(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平11－266161

(43)公開日　平成11年(1999)９月28日

(51)Int.Cl.[6]		識別記号	FI		
H03M	7/30		H03M	7/30	A
H04N	1/41		H04N	1/41	B
	7/30			7/133	Z

審査請求　未請求　請求項の数36　OL　外国語出願　（全 71 頁）

(21)出願番号　特願平10－274701

(22)出願日　平成10年(1998)９月29日

(31)優先権主張番号　PO9512

(32)優先日　1997年９月29日

(33)優先権主張国　オーストラリア（AU）

(71)出願人　591146745
キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテッド
CANON INFORMATION SYSTEMS RESEARCH AUSTRALIA PTY LTD
オーストラリア国 2113 ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ 1

(74)代理人　弁理士　大塚 康徳　(外２名)

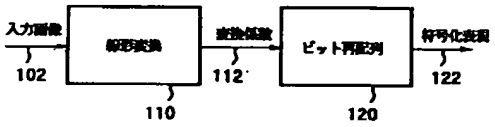
最終頁に続く

(54)【発明の名称】　データ圧縮方法及びデータ圧縮装置

(57)【要約】　（修正有）

【課題】ディジタルデータを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成し、対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化し、ディジタルデータ圧縮を行なう。

【解決手段】高レベルブロック図に示された入力画像１０２は望ましくは線形変換を行う変換ブロック１１０に提供されて、対応する変換係数１１２を生成する。ここでは離散・ウェイブレット変換（ＤＷＴ）、離散コサイン変換（ＤＣＴ）などを適用する。変換係数１１２、或いは、更に特定すれば、その値を表現するビットシーケンスは、符号化表現１２２を得るため、効率的な方法でビット再構成ブロック１２０により符号化される。



1

【特許請求の範囲】

【請求項1】ディジタルデータ圧縮方法であって、

（a）前記データを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成する変換ステップと、

（b）対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化する量子化ステップと、

を有し、

前記量子化ステップでは、前記クワッドツリーのリーフノードの各々が関連する量子化ファクタを有し、前記変換データの量子化に利用されることを特徴とするディジタルデータ圧縮方法。

【請求項2】前記クワッドツリーはレート歪みの意味における最適条件となるように決定されることを特徴とする請求項1に記載のディジタルデータ圧縮方法。

【請求項3】前記クワッドツリーは量子化ファクタのリストの前の、2進接頭表記を利用して符号化されることを特徴とする請求項1に記載のディジタルデータ圧縮方法。

【請求項4】ラグランジュ乗数法が、前記最適条件の決定に用いられることを特徴とする請求項2に記載のディジタルデータ圧縮方法。

【請求項5】前記最適条件はデータアイテムごとの所定のビットのために最適化されることを特徴とする請求項4に記載のディジタルデータ圧縮方法。

【請求項6】前記ディジタルデータは画像データを含むことを特徴とする請求項1に記載のディジタルデータ圧縮方法。

【請求項7】前記ディジタルデータは、動画データを含むことを特徴とする請求項1に記載のディジタルデータ圧縮方法。

【請求項8】前記動画データは、フレーム誤差データを含むことを特徴とする請求項7に記載のディジタルデータ圧縮方法。

【請求項9】ディジタルデータ伸張方法において、

該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、

（a）前記量子化及びクワッドツリー情報を復号する情報復号ステップと、

（b）前記符号化された量子化係数を復号する量子化係数復号ステップと、

（c）前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化ステップと、

（d）前記逆変換係数を逆変換するステップと、

を有し、

前記逆量子化ステップにおいて、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とするディジタルデータ伸張方法。

2

【請求項10】前記ディジタルデータは、画像データを含むことを特徴とする請求項9に記載のディジタルデータ伸張方法。

【請求項11】前記ディジタルデータは動画データを含むことを特徴とする請求項9に記載のディジタルデータ伸張方法。

【請求項12】前記動画データは、フレーム誤差データを含むことを特徴とする請求項11に記載のディジタルデータ伸張方法。

【請求項13】ディジタルデータ圧縮装置であって、

前記データを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成する変換手段と、

対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化する量子化手段と、

を有し、

前記量子化手段では、前記クワッドツリーのリーフノードの各々が、関連する量子化ファクタを有し、前記変換データの量子化に利用されることを特徴とするディジタルデータ圧縮装置。

【請求項14】前記クワッドツリーはレート歪みの意味における最適条件となるように決定されることを特徴とする請求項13に記載のディジタルデータ圧縮装置。

【請求項15】前記クワッドツリーは量子化ファクタのリストの前の、2進接頭表記を利用して符号化されることを特徴とする請求項13に記載のディジタルデータ圧縮装置。

【請求項16】ラグランジュ乗数法が、前記最適条件の決定に用いられることを特徴とする請求項14に記載のディジタルデータ圧縮装置。

【請求項17】前記最適条件はデータアイテムごとの所定のビットのために最適化されることを特徴とする請求項16に記載のディジタルデータ圧縮装置。

【請求項18】前記ディジタルデータは画像データを含むことを特徴とする請求項13に記載のディジタルデータ圧縮装置。

【請求項19】前記ディジタルデータは、動画データを含むことを特徴とする請求項13に記載のディジタルデータ圧縮装置。

【請求項20】前記動画データは、フレーム誤差データを含むことを特徴とする請求項19に記載のディジタルデータ圧縮装置。

【請求項21】ディジタルデータ伸張装置において、

該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、

前記量子化及びクワッドツリー情報を復号する情報復号手段と、

前記符号化された量子化係数を復号する量子化係数復号手段と、

前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化手段と、
前記逆変換係数を逆変換する手段と、
を有し、
前記逆量子化手段において、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とするディジタルデータ伸張装置。
【請求項２２】前記ディジタルデータは、画像データを含むことを特徴とする請求項２１に記載のディジタルデータ伸張装置。
【請求項２３】前記ディジタルデータは動画データを含むことを特徴とする請求項２１に記載のディジタルデータ伸張装置。
【請求項２４】前記動画データは、フレーム誤差データを含むことを特徴とする請求項２３に記載のディジタルデータ伸張装置。
【請求項２５】ディジタルデータ圧縮のためのコンピュータプログラムが記録されたコンピュータ可読媒体を含むコンピュータプログラム製品であって、
前記データを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成する変換手段と、
対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化する量子化手段と、
を有し、
前記量子化手段では、前記クワッドツリーのリーフノードの各々が、関連する量子化ファクタを有し、前記変換データの量子化に利用されることを特徴とするコンピュータプログラム製品。
【請求項２６】前記クワッドツリーはレート歪みの意味における最適条件となるように決定されることを特徴とする請求項２５に記載のコンピュータプログラム製品。
【請求項２７】前記クワッドツリーは量子化ファクタのリストの前の、２進接頭表記を利用して符号化されることを特徴とする請求項２５に記載のコンピュータプログラム製品。
【請求項２８】ラグランジュ乗数法が、前記最適条件の決定に用いられることを特徴とする請求項２７に記載のコンピュータプログラム製品。
【請求項２９】前記最適条件はデータアイテムごとの所定のビットのために最適化されることを特徴とする請求項２８に記載のコンピュータプログラム製品。
【請求項３０】前記ディジタルデータは画像データを含むことを特徴とする請求項２５に記載のコンピュータプログラム製品。
【請求項３１】前記ディジタルデータは、動画データを含むことを特徴とする請求項２５に記載のコンピュータプログラム製品。
【請求項３２】前記動画データは、フレーム誤差データを含むことを特徴とする請求項３１に記載のコンピュータプログラム製品。
【請求項３３】ディジタルデータ伸張のためのコンピュータプログラムが記録されたコンピュータ可読媒体を含むコンピュータプログラム製品において、
該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、
前記量子化及びクワッドツリー情報を復号する情報復号手段と、
前記符号化された量子化係数を復号する量子化係数復号手段と、
前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化手段と、
前記逆変換係数を逆変換する手段と、
を有し、
前記逆量子化手段において、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とするコンピュータプログラム製品。
【請求項３４】前記ディジタルデータは、画像データを含むことを特徴とする請求項３３に記載のコンピュータプログラム製品。
【請求項３５】前記ディジタルデータは動画データを含むことを特徴とする請求項３３に記載のコンピュータプログラム製品。
【請求項３６】前記動画データは、フレーム誤差データを含むことを特徴とする請求項３５に記載のコンピュータプログラム製品。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、特にデジタル画像圧縮に適用可能なコンピュータデータ圧縮の分野に関する。更に、本発明は、離散ウェイブレット変換した静止画及び動画のデータのクワッドツリーを用いた空間的適合的な量子化のためのものである。
【０００２】
【従来の技術】デジタルデータの圧縮、特にデジタル画像圧縮及びデジタル動画圧縮の分野は、ここの所大きな関心が寄せられてきている。
【０００３】デジタル画像圧縮分野では、多くの様々な技術が利用されている。特に、１つのよく知られた技術として、ＪＰＥＧ標準がある。これは、画像の標準的なサイズのブロックを対応する余弦成分に変換する、離散コサイン変換（ＤＣＴ）を利用するものである。この点において、より高い周波数の余弦成分は、実質的圧縮ファクタを得ることを補助するために大きく量子化される。そのような大幅な量子化は画像圧縮の「非可逆符号化（ｌｏｓｓｙ）」技術のひとつである。ＪＰＥＧ標準には、また、変換係数に対し続けて可逆符号化（ｌｏｓｓｌｅｓｓ）圧縮するものもある。

【0004】最近では、ウェイブレット変換の分野はデータ圧縮に代わる形式として大きな注目を集めている。ウェイブレット変換は、鋭いエッジなどの不連続性を有するデータを表現するのに非常に適していることが明らかにされている。そのような不連続性は画像データ等にしばしば存在する。

【0005】以下の本発明の好適な実施の形態には画像データの圧縮について記述するが、もちろん、好適な実施の形態は、それに限定されるものではない。信号に対するウェイブレット解析の様々な適用例として、１９９６年１０月版ＩＥＥＥスペクトラム２６頁～３５頁にあるブルースら著の「ウェイブレット解析」と題した調査文献が参考とできる。

【0006】コンピュータ・グラフィックスに対するウェイブレットの様々な適用法を論じたものとしては、Ｍｏｒｇａｎ　Ｋａｕｆｍａｎｎ　Ｐｕｂｌｉｓｈｅｒｓ，Ｉｎｃ．から１９９６年に出版されたＩ．　Ｓｔｏｌｌｉｎｉｔｚら著「コンピュータ・グラフィックスのためのウェイブレット」がある。

【0007】さらに、動画に適用できるディジタル圧縮技術もまたよく知られている。たとえば、ビデオフレーム誤差信号に依存する圧縮技術もまたよく知られている。

【0008】主観的にも客観的にも、ディジタル信号をより圧縮するためには、全信号で固定量の量子化を行うより、各領域で量子化の量を変化させたほうがよい。もちろん、その信号ごとに量子化をどれほど変化させるかを示すのに、しばしばいくつかの符号化オーバーヘッドが必要となる。

【0009】

【発明が解決しようとする課題】従来から、適合的量子化は、例えば、動画圧縮標準ＭＰＥＧやその様々な実行に用いられてきた。

【0010】現在知られている固定技術よりももっと適合的な方式で信号の量子化を行なうことが望まれている。

【0011】

【課題を解決するための手段】上記目的を達成するため、本発明にあっては、ディジタルデータ圧縮方法であって、（ａ）前記データを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成する変換ステップと、（ｂ）対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化する量子化ステップと、を有し、前記量子化ステップでは、前記クワッドツリーのリーフノードの各々が関連する量子化ファクタを有し、前記変換データの量子化に利用されることを特徴とする。

【0012】また、ディジタルデータ伸張方法において、該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、（ａ）前記量子化及びクワッドツリー情報を復号する情報復号ステップと、（ｂ）前記符合された量子化係数を復号する量子化係数復号ステップと、（ｃ）前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化ステップと、（ｄ）前記逆変換係数を逆変換するステップと、を有し、前記量子化ステップにおいて、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とする。

【0013】また、本発明にあっては、ディジタルデータ圧縮装置であって、前記データを、離散ウェイブレット変換を利用して変換し、対応する変換データを生成する変換手段と、対応するクワッドツリー構造によって決定された可変量子化方式を利用して前記変換データを量子化する量子化手段と、を有し、前記量子化手段では、前記クワッドツリーのリーフノードの各々が、関連する量子化ファクタを有し、前記変換データの量子化に利用されることを特徴とする。

【0014】また、ディジタルデータ伸張装置において、該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、前記量子化及びクワッドツリー情報を復号する情報復号手段と、前記符合された量子化係数を復号する量子化係数復号手段と、前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化手段と、前記逆変換係数を逆変換する手段と、を有し、前記量子化手段において、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とする。さらに本発明は、ディジタルデータ伸張のためのコンピュータプログラムが記録されたコンピュータ可読媒体を含むコンピュータプログラム製品において、該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、前記量子化及びクワッドツリー情報を復号する情報復号手段と、前記符合された量子化係数を復号する量子化係数復号手段と、前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量子化手段と、前記逆変換係数を逆変換する手段と、を有し、前記量子化手段において、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とする。また、ディジタルデータ伸張のためのコンピュータプログラムが記録されたコンピュータ可読媒体を含むコンピュータプログラム製品において、該ディジタルデータは、符号化された量子化係数と、関連する量子化及びクワッドツリー情報であって、前記量子化及びクワッドツリー情報を復号する情報復号手段と、前記符合された量子化係数を復号する量子化係数復号手段と、前記量子化及びクワッドツリー情報に従って、前記符号化された量子化係数を逆量子化する逆量

子化手段と、前記逆変換係数を逆変換する手段と、を有し、前記量子化手段において、各クワッドツリーのリーフノードは、前記復号された量子化の逆量子化に用いられる関連量子化ファクタを含むことを特徴とする。

【００１５】

【発明の実施の形態】以下にステップ及び／又は特徴部分が表された図面を参照して、この発明の好適な実施の形態を例示的に詳しく説明する。これらのステップ、特徴部分は、反対の意図が示されない限り、同一の参照符号が付された場合には、同一の機能及び／又は動作をすることを示しているものである。

【００１６】以下に好適な実施の形態を説明する前提として、キャノン　インフォメーション　システムズ　リサーチ　オーストラリア　Ｐｔｙ．Ｌｔｄ．により１９９７年１月２２日に出願された「デジタル画像圧縮」と題したオーストラリア仮特許出願Ｎｏ．ＰＯ４７２８において開示された画像圧縮／伸張方法が存在する。

【００１７】この圧縮／伸張方法について、（ＳＷＥＥＴ画像圧縮方法の概要）、（第１のＳＷＥＥＴ画像圧縮方法の符号化処理）、（第１のＳＷＥＥＴ画像圧縮方法の復号処理）、（２次元的な例）、そして（第２のＳＷＥＥＴ画像圧縮方法の符号化処理）と表題を付けた節に分けて説明する。

【００１８】（ＳＷＥＥＴ画像圧縮方法の概要）この符号化方法に係る実施の形態の概要を説明するため、その高レベルブロック図を図１に示す。入力画像１０２は望ましくは線形変換を行う変換ブロック１１０に提供されて、対応する変換係数１１２を生成する。ここでは離散・ウェイブレット変換（ＤＷＴ）を用いてもよい。

【００１９】画像の２次元ＤＷＴとは、画像に対する一つの低周波近似成分と３つの高周波細部成分を用いて画像を表現する変換である。一般的に、これらの成分はサブバンドと呼ばれている。ＤＷＴで形成されたこれら４つのサブ画像の各々は、もとの画像のサイズの１／４である。低周波画像はもとの画像に関する情報の大半を占める。この情報、またはエネルギー・コンパクションは、画像圧縮に活用される離散ウェイブレット変換画像サブバンドの特徴である。

【００２０】単一レベルＤＷＴは、その低周波画像、またはサブバンドに対して繰り返し任意の回数だけ適用できる。例えば、画像の３レベルＤＷＴは、１回変換を適用してから変換によって得られた低周波サブバンドにさらにＤＷＴを適用することで得られる。つまり、これにより９個の細かい（detail）サブバンドと１つの（非常に）低周波のサブバンドが得られる。３レベルＤＷＴの後でも、得られた低周波サブバンドはもとの画像についての相当量の情報を含んでおり、なおかつ６４分の１の小ささ（１／４×１／４×１／４）であることから、圧縮において係数６４となる。

【００２１】しかし、画像データを非相関させるための他の線形変換も実行できる。例えば、離散コサイン変換（ＤＣＴ）も適用できる。変換係数１１２、或いは、更に特定すれば、その値を表現するビットシーケンスは、符号化表現１２２を得るため、効率的な方法でビット再構成ブロック１２０により符号化される。

【００２２】復号処理は符号化処理の単なる逆である。符号化係数は変換係数に復号される。（変換ドメインの）画像は逆変換されてもとの画像、またはそのある程度の近似を形成する。

【００２３】本発明の実施の形態について更なる説明を進める前に、以下で用いる術語の概略を説明する。二進整数表現において、数「ビットｎ」または「ビット番号ｎ」とは、２進数の桁で最下位ビットから左側にｎ番目を表わす。例えば、８ビットのバイナリ表現を仮定すると、十進数９は００００１００１で表わされる。この数で、ビット３は１にあたり、ビット２，ビット１，ビット０は各々０，０，１にあたる。更に、変換は複数の列及び行で構成された係数マトリクスとして表わされ、それぞれの係数はビットシーケンスによって表わされる。概念的にマトリクスと言う場合には、３つの次元が含まれている。列方向の第１次元、行方向の第２次元、ビットシーケンス方向の第３次元の３つである。同じビット番号で各ビットシーケンスを通過する３次元空間における１平面はビットプレーンと呼ばれる。

【００２４】変換符号化アプリケーションのため、係数の可能な範囲を表わすために必要とされる係数あたりのビット数は、線形変換と入力画素の（画素あたりビット数で）各画素の解像度により決定される。各画素の値の範囲は代表的には変換係数の大半の値に対して大きく、大半の係数はゼロが先行する大きな数を有する。例えば、数値９は８ビット表現で４個のゼロが先行し、１６ビット表現では１２個のゼロが先行する。本圧縮方法及び圧縮装置は、係数のブロックに対して、これらの先行するゼロを、効率的な方法で表現する（または符号化する）。残りのビットと数値の符号は変更なしに直接符号化される。

【００２５】簡単にいうと、不必要に本発明を不明瞭にしないために、以下には符号を示す１ビットを有する変換係数も、符号のない２進数の形で表現されるものと仮定する。つまり、十進数－９と９は同じビット列即ち１００１で表わされているが、前者は負の値であることを表わす符号ビット１を有し、後者は正の値を表わす符号ビット０を有しているものとする。先行するゼロの個数は変換係数のレンジによって決まる。整数表現を用いる際には、係数はすでに暗黙のうちに最も近い整数値へ量子化されるが、これは本発明においては必ずしも必要ではない。さらに、圧縮するため、小数点以下のビットに含まれる何らかの情報は通常無視される。

【００２６】領域は一組の連続的な画像係数で構成される。係数という述語は、以下の説明では画素と相互交換

可能なように使用されるが、当業者には良く理解されるように、前者は変換ドメイン（例えばＤＷＴドメイン）内の画素を表わすために用いられるのが代表的である。

【００２７】（第１のＳＷＥＥＴ画像圧縮方法の符号化処理）図３及び図４を参照して、第１の画像圧縮方法について更に詳細に説明する。

【００２８】図３はこの画像符号化方法を示すフローチャートである。ステップ３０２において、入力画像を用いて処理が開始される。ステップ３０４で、入力画像は線形変換望ましくは離散・ウェイブレット変換を用いて変換される。最初の領域は画像全体となるように定義される。例えば、入力画像の３レベルＤＷＴの場合、得られる係数は１０個のサブバンドから構成され領域として指定できる。これ以外に、各々のサブバンドを別々に処理し、各々の初期領域を問題とするサブバンド全体に設定することができる。

【００２９】ステップ３０６では、絶対値が最大の変換係数の最上位ビット（ＭＳＢ）が決定され、パラメータmaxBitNumberがこの係数値にセットされる。例えば、最大の変換係数が二進数の値００００１００１（十進数９）の場合、パラメータmaxBitNumberは３にセットされるが、これはＭＳＢがビット番号３であるためである。これ以外に、パラメータmaxBitNumberは絶対値が最大の変換係数のＭＳＢより大きい何らかの値となるようにセットしても良い。

【００３０】さらに、ステップ３０６で、符号化パラメータminBitNumberが符号化画像品質を指定するようにセットされる。さらに詳しくは、この符号化パラメータは、変換される画像の全ての係数の精度を指定し、必要に応じて変更できる。例えば、minBitNumberが３では、値が１より元の画像の再現がより粗くなる。

【００３１】オプションとして、本技術が入力画像の符号化表現で出力ヘッダを提供するステップ３０８を含む。こうすれば、実際の実行の際、ヘッダ情報は符号化表現の一部として出力される。例えば、出力ヘッダは、画像の高さと幅、ＤＷＴのレベル数、ＤＣサブバンドの平均値、パラメータmaxBitNumber、パラメータminBitNumberを含むソース画像についての情報を含む。

【００３２】ステップ３１０が始まると、変換画像の各々のサブバンドがステップ３１２とステップ３１４で別々に符号化される。各サブバンドは独立して、低周波から高周波の順番に符号化される。ＤＣサブバンドでは、符号化の前に平均値が除去されステップ３０８でヘッダ情報に符号化される。ステップ３１２では、各々のサブバンドはサブバンド全体として初期領域をセットすることにより符号化される。ステップ３１４では、パラメータとしてmaxBitNumberおよびminBitNumberにより領域が符号化される。これは、画像の低解像度バージョンが高解像度以前にビットストリームに符号化されるため階層化コードを提供する。処理はステップ３１６で終了する。

【００３３】図４は各々の領域を符号化するために図３のステップ３１４でコールされる手順"Code region(currentBitNumber,minBitNumber)の詳細なフローチャートである。ここでmaxBitNumberがcurrentBitNumberとして提供される。ステップ４０２で処理が始まる。図４の領域符号化処理への入力はcurrentBitNumberとminBitNumberパラメータを含む。望ましくは、本方法は、選択された領域または部分領域で処理がそれ自体をコールできるような再帰的技術として実行される。しかし、処理は非再帰的な方法で実行してもよい。

【００３４】決定ブロック４０４で、currentBitNumberパラメータがminBitNumberパラメータより小さいか判定するチェックを行なう。それ以外に、決定ブロック４０４が真（イエス）を返す場合には何も行なわずに処理はステップ４０６の呼び出し手順に戻る。この条件は、選択された領域のあらゆる係数がminBitNumberより小さいＭＳＢ番号を有することを表わしている。決定ブロック４０４が偽（ノー）を返す場合、処理は決定ブロック４０８に進む。

【００３５】決定ブロック４０８では、選択された領域が１×１画素か調べるチェックを行なう。決定ブロック４０８が真（イエス）を返す場合、処理はステップ４１０に進む。ステップ４１０では、１×１画素が符号化される。望ましくは、これは符号化表現にminBitNumberより上の残りのビットを直接出力することからなる。ステップ４１２では、処理は呼び出し手順に戻る。それ以外の場合、決定ブロック４０８が偽（ノー）を返す場合、領域は１つ以上の係数から構成されており処理は決定ブロック４１４に続く。

【００３６】決定ブロック４１４では、有意かどうか決定するために選択領域をチェックする。つまり、領域の有意性を調べる。領域内で各々の係数のＭＳＢがcurrentBitNumberパラメータの値より小さい場合には領域は有意でないとされる。領域有意性の考え方を正確にするため、式（１）に数学的定義を掲載する。任意のビット番号で、仮にcurrentBitNumber＝ｎで、領域は次のような場合に有意でないとされる：

【００３７】

【数１】

$$|c_{ij}| < 2^n, \forall i,j \in R,$$

ここでＲは領域、またｃijはこの領域内の係数(i,j)を表わす。

【００３８】決定ブロック４１４が偽（ノー）を返した場合、処理はステップ４１６に続く。ステップ４１６では、値０（すなわち第１のトークン）が符号化表現ストリームに出力され、currentBitNumberパラメータが１だけデクリメントされる。つまり、この領域の次に低いビットプレーンが処理のために選択される。処理は決定ブ

ロック４０４に続き、ここでパラメータcurrentBitNumber-1とminBitNumberによりその領域が再処理される。それ以外の場合、決定ブロック４１４が真（イエス）を返した場合、つまり領域が有意な場合、処理はステップ４１８に続く。

【００３９】ステップ４１８では、値１（すなわち第２のトークン）が符号化表現ストリームに出力される。ステップ４２０では、指定した分割アルゴリズムを用いて選択領域が所定数（望ましくは４）の部分領域に分割される。使用する分割アルゴリズムでコーダに分かっている。

【００４０】本方法では正方形領域を使用する。領域は４個の等しいサイズの（正方形）部分領域に分割されるのが望ましい。図２に図示してあるように、選択領域（Ｒ）２００はＭ×Ｍ係数のサイズを有し４つの等しいサイズの部分領域２１０，２１２，２１４，２１６に分割される。部分領域の各々はＮ×Ｎのサイズで、ＮはＭ／２に等しい。初期領域のサイズと形状によってはこれが常に可能になるわけではない。不可能な場合には、初期領域を多数の正方形領域に分割し、各々が２のべき乗の寸法を取るようにし、これらの区画を別々に符号化できる。いずれの場合にも、インテリジェントな方法で行なわれれば総合的な結果に対してこの初期化がもたらす影響は最少である。別の方法として、ブロック単位のコーダに適している異なる分割を用いてもよい。

【００４１】ステップ４２２では、各々の部分領域が同じcurrentBitNumberとminBitNumberパラメータで符号化される。これは図４の手順 "Code region (currentBitNumber,minBitNumber)"の再帰的呼び出しを用いて行なうのが望ましい。部分領域の符号化は並列にまたは逐次的に実施できる。後者の場合、処理は低周波サブバンドから高周波サブバンドへ向かって始める。

【００４２】符号化表現において、変換係数はcurrentBitNumberからminBitNumberへ画素ビットを単純に出力することで符号化する。望ましくは、標記にしたがい係数ビットの幾つかが非ゼロの場合にのみ符号を出力する。例えば、currentBitNumber=3の場合で、minBitNumber=1であれば、－９（００００１００１）は符号ビット「１」に続く「１００」に符号化される。

【００４３】（第１のＳＷＥＥＴ画像圧縮方法の復号処理）図５は、図３および図４の処理を用いて得られた画像の符号化表現を復号する方法を示すフローチャートである。ステップ５０２において、符号化表現を用いて処理を開始する。ステップ５０４では、ヘッダ情報が符号化表現から読み取られてもとの画像のサイズを決定し、これによって初期領域サイズを決定する。また、maxBitNumber（符号化処理における初期のcurrentBitNumberに等しい）やminBitNumberなどの情報が入力される。更なる情報としてはＤＣサブバンドの平均値などが挙げあれる。

【００４４】ステップ５０６では、各サブバンドの復号が各々のサブバンドへ領域を設定することから開始される。ステップ５０８では、maxBitNumberとminBitNumberパラメータを用いて選択領域が復号される。ステップ５１０では、逆ＤＷＴを復号した選択領域にて起用する。処理はステップ５１２で終了する。

【００４５】図６は手順コール"Decode region(currentBitNumber, minBitNumber)を用いて各領域を符号化するための図５のステップ５０８の詳細なフローチャートである。ここでmaxBitNumberはcurrentBitNumberとして提供される。ステップ６０２で処理が始まる。図６の領域復号処理への入力はcurrentBitNumberとminBitNumberパラメータである。また、本方法は再帰的技術として実行するのが望ましい。しかし、この処理は非再帰的な方法で実現することもできる。

【００４６】決定ブロック６０４では、currentBitNumberがminBitNumberより小さいか決定するチェックを行なう。決定ブロック６０４が真（イエス）を返す場合、処理はステップ６０６に続き、ここで呼び出し手順に処理が復帰する。それ以外の場合、決定ブロック６０４が偽（ノー）を返した場合、処理は決定ブロック６０８に続く。

【００４７】決定ブロック６０８では、選択領域が１×１画素のサイズを有するか決定するチェックを行なう。決定ブロック６０８が真（イエス）を返す場合、処理はステップ６１０に続く。ステップ６１０では、１×１領域が復号される。処理はステップ６１２で呼び出し手順に戻る。決定ブロック６０８が偽（ノー）を返した場合、処理はステップ６１４に続く。ステップ６１４では、符号化表現からビットが入力される。

【００４８】決定ブロック６１６では、ビットが１に等しいか決定するチェックを行なう。即ち領域が有意か決定するために入力をチェックする。決定ブロック６１６が偽（ノー）を返した場合処理はステップ６１８に進む。ステップ６１８では、currentBitNumberが決定され、処理は決定ブロック６０４に続く。それ以外の場合、決定ブロック６１６が真（イエス）を返した場合、処理はステップ６２０に進む。ステップ６２０では、領域が所定の個数（望ましくは４）の部分領域に分割される。ステップ６２２では、部分領域の各々がcurrentBitNumberとminBitNumberを用いて復号される。これは図６に図示してある処理への再帰的呼び出しを用いて行なわれる。ステップ６２４で処理は呼び出し手順に戻る。

【００４９】このように、エンコーダでの有意性の決定から出力されたビットがアルゴリズムのどのパスを取るかデコーダに指示し、こうしてエンコーダに倣う。画素とおそらくは符号も適当なビット数（currentBitNumberからminBitNumberまで、及び、それらに非ゼロがある場合には符号ビット）で単純に読み出すことにより復号される。

【００５０】（２次元的な例）本方法は大半の変換係数の先行するゼロを効率的に符号化し、一方で最上位ビットから所定の最下位ビットへ、パラメータminBitNumberにより指定されるビットを符号化し、符号は単純にそのままとする。このようにして、本圧縮方法ではうまく先行するゼロを表現している。本方法はある状況で、即ち離散・ウェイブレット変換画像係数の符号化で非常に有効であり、代表的には広いダイナミックレンジが示される。代表的には少数の係数が非常に大きな値を有しているが、大半は非常に小さい値を有している。

【００５１】４×４係数を含む2次元領域の符号化の例について、図7（ａ）から図7（ｄ）を参照して説明する。図7（ａ）の4×4領域７００の処理は、係数全部のうちで最大のビット番号（ビットプレーン）である7にmaxBitNumberをセットして開始する。

【００５２】

【数2】

$$\begin{bmatrix} 200 & 13 & -11 & -8 \\ -13 & 3 & -4 & -3 \\ 8 & 1 & -2 & -2 \\ 2 & -1 & -3 & -3 \end{bmatrix}.$$

図示の目的では、minBitNumberは3にセットされる。ヘッダはmaxBitNumberとminBitNumberを含む符号化表現に出力されるのが望ましい。領域７００を符号化する処理は次のように進む。

【００５３】currentBitNumber＝7で、領域７００はビット番号7（図4の決定ブロック４０４，４０８，４１４およびステップ４１８を参照）に対して有意であるから出力される。領域７００は、図7（ａ）の左上の領域７１０，右上の領域７１２，左下の領域７１４，右下の領域７１６の4個の部分領域に分割される（図4のステップ４２０参照）。部分領域各々は2×2係数で構成される。

【００５４】図7（ａ）の部分領域７１０，７１２，７１４，７１６はさらに図7（ｂ）に図示してある所定の処理シーケンスで符号化される。領域７５０は4個の部分領域７５０Ａ～７５０Ｄで構成される。図面に示してある3本の矢印は処理の順番またはシーケンス、即ち左上の部分領域７５０Ａ、右上の部分領域７５０Ｂ、左下の部分領域７５０Ｃ、右下の部分領域７５０Ｄそれぞれの処理を表わす。

【００５５】図7（ａ）の部分領域７１０は最初に符号化される（図4のステップ４２２参照）。currentBitNumberが7に等しい場合、1が符号化表現に出力される。部分領域７１０は十進数の値２００，１３，－１３，３を有する4個の1×1画素に分割される。これらの係数の各々はcurrentBitNumber＝7からminBitNumber＝3までの各係数のビットを出力することで符号化される（図4の決定ブロック４０８とステップ４１０参照）。符号ビットは必要なら出力される。このようにして、十進数での値が２００なら符号ビット0に続けて１１００１と符号化される。係数値１３は符号ビット0付きで００００１として符号化される。係数値－１３では符号ビット1のついた００００１に符号化される。最後に、係数値3は００００００（符号ビットなし）に符号化される。各係数の符号化表現は、currentBitNumberとminBitNumberの間に、係数「２００」のビットに先行する２個の「１」ビットを含む。これで左上の部分領域７１０の符号化が完了する。この状態での符号化出力は次のようになる：

【００５６】

【数3】

$$\underbrace{1111001}_{200}\ \overset{\text{sign bit}}{\underbrace{0}}\ \underbrace{00001}_{13}\ \underbrace{00001}_{-13}\ \underbrace{100000}_{3}.$$

ヘッダ情報は前述の表現に示していない。

【００５７】右上の部分領域７１２が次に符号化される（図7（ｂ）参照）。領域７１２は、7，6，5，4に等しいcurrentBitNumberの各々について有意でないので、これらのビット番号で0が出力される。currentBitNumber＝3では1が出力されるが、これはビットプレーンがビット番号3に対して有意であるためである。部分領域７１２は値－１１，－８，－４，－３を有する1×1画素4個に分割される。これらの十進数の値は、符号ビット1のついたビット値1，符号ビット1のビット値1，符号ビットなしのビット値0，0に各々符号化される。つまりこの段階で、符号化表現は次のようになる：

【００５８】

【数4】

$$1111001\underbrace{0}_{\text{sign bit}}00001\underbrace{0}_{-11}00001\underbrace{1}_{-8}100000\underbrace{1}_{-4}\underbrace{0}_{-3}.$$

左下の部分領域７１４が次に符号化される。領域７１４は、7，6，5，4に等しいcurrentBitNumberの各々について有意でないので、これらのビット番号で0が出力される。currentBitNumber＝3では1が出力されるが、これはビットプレーンがビット番号3に対して有意であるためである。部分領域７１４は値8，1，2，－1を有する1×1画素4個に分割される。これらが各々、符号ビット0の2進数1、および符号ビットのない2進数0，0，0に符号化される。

【００５９】最後に、値－2，－2，－3，－3を有する右下の部分領域７１６が符号化される。currentBitNumber＝7，6，5，4，3の各々で部分領域７１６がこれらのビット数に対して有意ではないため0を出力する。符号ビットは出力されない。つまり符号化表現は次

のようになる：

【0060】

【数5】

111100100000100000110000000001111100000011000000000.

デコーダは単純に符号化処理にならって図7（c）に図示したように符号化表現から領域を再構成する。

【0061】復号処理は多数の方法で「よりスマート」に行なうことができる。このような「よりスマート」な方法の1つが図7（d）に図示してある。この場合、非ゼロ係数の大きさは2のminBitNumberのべき乗の半分ずつ各々増加する。これが図7（d）に図示してある。このようにすると、「スマート」な復号処理は、復号した係数ともとの係数の間の平均二乗誤差を一般的に減少できる。さらに、エンコーダはこれ以外にもこの（種の）演算を実行でき、これによってデコーダに図7（c）に図示した最も簡単な方法を利用させる。

【0062】（第2のSWEET画像圧縮方法の符号化処理）他のSWEET法による符号化処理について、図9から図12を参照して説明する。

【0063】デジタル画像全体の離散・ウェイブレット変換はブロック単位で実行できる。各ブロックでの変換の結果は一組の係数となり、これは基本的に画像全体の離散・ウェイブレット変換の一組の空間的に対応する係数と等しい。例えば、画像全体についての離散・ウェイブレット変換の所定の係数の組から、デジタル画像の一部またはブロックを指定された細部まで再現できる。周波数ドメインから所定の係数の組を選択することは実質的に空間ドメインからのデジタル画像（ブロック）の対応する部分を表現することになる。デジタル画像のブロック単位の離散・ウェイブレット変換は、画像を複数ブロックに分割し各ブロックに対して独立して変換を適用することにより実行でき、これによって実質的に現在の空間的な位置に関連したDWT係数を評価することができる。ブロック単位の変換アプローチを採用する利点は、画像の他のブロックとの最小限の相互作用で（実質的に独立して）ブロックを実質的に符号化できることである。ブロック単位の技術は本質的にメモリーにローカライズされており、そのためコンピュータ・システムを用いて実現した場合には一般に効率的である。

【0064】図9は第2の符号化方法のブロック単位の符号化処理を示すフローチャートである。処理はステップ902から始まる。ステップ904で、ヘッダが出力される。この情報は画像の高さと幅、DWTのレベル数、2個のパラメータmaxBitNumberとminBitNumberを含むのが望ましい。オプションとして、用途によっては多少なりともヘッダ情報を使用できる。

【0065】符号化パラメータmaxBitNumberは様々な方法で選択できる。全ての画像ブロックについて、これらのうちのどれかを符号化する前にブロックDWTを実行する場合、maxBitNumberは全部のDWTブロックにわたる最大の係数のMSB番号となるように選択できる。例えば、最大の係数が10000001（十進数129）だとすると、maxBitNumberはMSBがビット番号7であるため7にセットされる。これ以外に、入力画像の変換と解像度によって決まる決定閾値を用いることができる。例えば、8ビット入力画像（7ビットと符号にレベルがシフトされる）およびハール変換では、最大のMSBはJ＋7で区切られ、JはDWTのレベル数である。ブロックが小さい場合、このパラメータの選択は圧縮に対して有意な影響を有することがある。場合によっては、maxBitNumberを選択するもっと洗練された方法を用いることがある。しかし、これは特定のアプリケーションに依存する。

【0066】パラメータminBitNumberは圧縮比に対する画質の兼ね合いを決定し変更できる。例えば、ほぼ直交に近い変換では、値3が8ビットグレースケールまたは24ビットRGB画像で充分な画質を提供する。

【0067】ステップ906では、画像がブロックに分解される（または画像ブロックが形成される）。画像は重複するブロックに分割されるのが望ましい。しかし、重複しないブロックを用いることもある。係数のブロックはもとの画像全体と同程度の大きさとするか、または8×8係数（3レベル変換で）と同程度の小ささにできる。メモリーの少ないアプリケーションでは、できる限り小さいブロックを使用する。一般に、16係数のブロックサイズが、3乃至4レベルDWTによる高レベルの圧縮に充分である。3レベルDWTでの8×8係数のブロックサイズは各ブロックのDC係数に対して差動パルス符号変調（DPCM）を用いることで良好な符号化効率を維持できる。

【0068】ステップ908で、各ブロックはレベルシフトされ変換が実行される。望ましくは、DWTが使用される。画像値はレベルシフトされ（例えば、8ビット画像では128だけシフトされる）、望ましくない平均バイアスを減少または排除し、画像の各々の空間ブロックが変換される。DWTでは、現在のブロックを包囲するブロックについての何らかの知識が必要とされるのが普通だが（また逆DWTでも同様だが）、これは厳密には必要とされない。

【0069】ステップ910では、maxBitNumberとminBitNumberパラメータを用いてブロックを符号化する。処理はステップ912で終了する。

【0070】ブロックを符号化するステップ910が図10のフローチャートに詳細に図示してある。図10のブロック符号化処理への入力はcurrentBitNumberおよびminBitNumberパラメータを含む。図9のステップ810において、maxBitNumberはcurrentBitNumberパラメータとして入力される。処理はステップ1002で始まる。決定ブロック1004では、currentBitNumberがminBitNumberより小さいか決定するチェックを行なう。決定ブ

ロック１００４が真（イエス）を返した場合、処理はステップ１００６に進む。ステップ１００６では、実行が呼び出し処理に返され、これによってブロック内の全ての係数がminBitNumberより小さいＭＳＢ番号を有することを表わす。それ以外の場合で、決定ブロック１００４は偽（ノー）を返すと、処理は決定ブロック１００８に進む。

【００７１】決定ブロック１００８では、現在のブロックが有意か決定するチェックを行なう。決定ブロック１００８が偽（ノー）を返した場合、処理はステップ１０１０に進む。ステップ１０１０では、符号化表現に0が出力されてcurrentBitNumberがデクリメントされ、言い換えれば次に低いビットプレーンを選択する。処理は決定ブロック１００４に進む。これ以外の場合、決定ブロック１００８が真（イエス）を返した場合、処理はステップ１０１２に進む。

【００７２】ステップ１０１０と併せて決定ブロック１００４と１００８により処理はブロック内で最大の係数のＭＳＢ番号を見つけ出すことができるようになる。ブロック内の全ての係数のＭＳＢ番号がcurrentBitNumberより小さければcurrentBitNumberに対してブロックは有意でない。これは、ブロックのビットプレーンが有意かまたはminBitNumberよりcurrentBitNumberが小さくなるまで反復される。

【００７３】ステップ１０１２で、ビットプレーンが有意であることを表わすように、符号化表現には１が出力される。ステップ１０１４では、ＤＣサブバンドが符号化される。ステップ１０１６では、ブロックの細部が、パラメータＪ，currentBitNumber，minBitNumberを用いて符号化される。ステップ１０１８では、実行が呼び出し手順に戻る。つまり、ブロックが有意であれば、ステップ１０１２，１０１４，１０１６が実行され、（一般化）クワドツリー・セグメンテーションを用いてminBitNumberより大きなＭＳＢ番号を有する全ての係数を見付けようとする。ブロックが有意な場合には、ＤＣサブバンド係数と残りの係数から構成されレベルＪに対して「ブロック細部」と呼ばれるブロックの、２個の「サブブロック」に分割される。これは全ての低いレベルでレベルＪのブロックについての高周波情報を表わすためである。

【００７４】ＤＣサブバンド符号化についての図１０のステップ１０１４が図１２のフローチャートで詳細に図示してある。つまり、図１２はcurrentBitNumberとminBitNumberパラメータを使用するサブバンドまたはサブブロックの符号化処理を示している。ステップ１２０２で処理が始まる。決定ブロック１２０４では、currentBitNumberがminBitNumberより小さいか判定するチェックを行なう。決定ブロック１２０４が真（イエス）を返したなら、処理はステップ１２０６に続く。ステップ１２０６では、実行が呼びだし手順に戻る。それ以外の場合、決定ブロック１２０４が偽（ノー）を返した場合、処理は決定ブロック１２０８に進む。

【００７５】決定ブロック１２０８では（サブバンドの）ブロックサイズが１×１画素か調べるチェックを行なう。決定ブロック１２０８が真（イエス）を返した場合、処理はステップ１２１０に進む。ステップ１２１０では、１×１画素が符号化される。これは、必要なら符号ビットに続けて、currentBitNumberとminBitNumberを含め、これらの間にあるビットを出力することによる。処理はステップ１２１２で呼びだし手順に戻る。それ以外の場合、決定ブロック１２０８が偽（ノー）を返したなら、処理は決定ブロック１２１４に進む。

【００７６】決定ブロック１２１４では、（サブバンドの）ブロックが有意か調べるチェックを行なう。決定ブロック１２１４が偽（ノー）を返した場合、処理はステップ１２１６に進む。ステップ１２１６では、符号化表現に0が出力されcurrentBitNumberがデクリメントされる。処理は決定ブロック１２０４に続く。それ以外の場合で、決定ブロック１２１４が真（イエス）を返したなら、処理はステップ１２１８に進む。

【００７７】ステップ１２１８では、（サブバンド）ブロックが有意であることを表わすよう１が符号化表現に出力される。ステップ１２２０では、（サブバンドの）ブロックが４個のサブブロックに分割される。ステップ１２２２では、図３２の処理に再帰的呼び出しを行ない、currentBitNumberとminBitNumberを用いて各々のサブブロックを符号化する。ステップ１２２４では、実行が呼び出し手順に戻る。

【００７８】つまり図１２に図示した処理では、サブバンドまたはそのサブブロックが符号化される。最大のＭＳＢ番号は前述のように分離される。サブブロックが１つの画素だけで構成される場合、単一の係数として符号化される。それ以外の場合、currentBitNumberがデクリメントされて、currentBitNumberがminBitNumberより小さくなるまで、またはサブバンド（サブブロック）が有意になるまで符号化表現には0が出力される。サブバンド（サブブロック）が有意な場合、４個の（できるだけ等しくなるように）サブブロックに分割され、これらがさらに符号化される。単一の係数、例えばＤＣ係数は、currentBitNumberからMINBINNへ係数ビットを出力することにより符号化される。また、符号は係数ビットの幾つかが非ゼロの場合にだけ出力するのが望ましい。

【００７９】ブロック細部を符号化するための図１０のステップ１０１６は図１１のフローチャートに図示してある。ステップ１１０２で処理が始まる。決定ブロック１１０４では、currentBitNumberがminBitNumberより小さいか判定するチェックを行なう。決定ブロック１１０４が真（イエス）を返した場合、ステップ１１０６で実行は呼び出し手順に戻る。それ以外の場合、決定ブロック１１０４が偽（ノー）を返したならには、処理は決定

ブロック１１０８へ進む。

【００８０】決定ブロック１１０８では、ブロック（細部）が有意か判定するチェックを行なう。決定ブロック１１０８が偽（ノー）を返した場合、処理はステップ１１１０に進む。ステップ１１１０では、符号化表現に0が出力されcurrentBitNumberがデクリメントされる。処理は決定ブロック１１０４に進む。それ以外の場合には、決定ブロック１１０８が真（イエス）を返したなら、処理はステップ１１１２に進む。

【００８１】ステップ１１１２では、ブロック（細部）が有意であることを表わすよう符号化表現に１が出力される。ステップ１１１４では、ＨＩＧＨ－ＬＯＷ（ＨＬ）、ＬＯＷ－ＨＩＧＨ（ＬＨ）、ＨＩＧＨ－ＨＩＧＨ（ＨＨ）周波数サブバンドの各々が符号化される。各々の解像度のＨＬ，ＬＨ，ＨＨ周波数サブバンドは共通にＡＣサブバンドと呼ばれる。これらのサブバンドの各々は図６の処理にしたがって符号化される。ステップ１１１６では、（ブロック細部が存在する場合）図５に図示してある処理への再帰的呼び出しにより、パラメータＪ－１，currentBitNumber，minBitNumberを用いてブロック細部が符号化される。ステップ１１１８で実行は呼び出し手順に戻る。

【００８２】このように、レベルＪでのブロック細部は最大の係数のＭＳＢ番号を最初に分離するように処理される。これはcurrentBitNumberをデクリメントしてブロックが有意になるまでゼロを出力することにより行なう。ブロックは次にレベルＪでの３個の高周波サブバンドとレベルＪ－１について（Ｊ－１が０より大きい場合）ブロック細部に分割される。この分割アプローチはいわゆる１／ｆ形式のスペクトルモデルにより誘導される。

【００８３】第２の圧縮方法に応じた復号処理は図９から図１２を参照して説明した符号化処理に倣うことにより実現できる。

【００８４】従って、この符号化及び復号方法及びその装置は、画像を記憶するか送信するかのいずれか或いは両方のために表現が適しているような効率的かつ柔軟性のある方法でデジタル画像データを表現する。符号化技術は一般に変換係数のアレイを表わすために使用でき、また離散・ウェイブレット変換ドメインにおいて画像を表現することにより効率的な表現を提供するために使用できる。特に、本方法及び装置は入力画像から取得した変換係数のブロックの先行するゼロを表現する（または符号化する）ものである。本技術は、任意のサイズのコードで元の画像の良好な再現を与える点に関してと、高速な復号処理を提供する点に関して効率的である。さらに、本技術は線形変換から得られた係数がエントロピー符号化を使用することなく独立して符号化される点で柔軟性がある。実施の形態の有利な側面としては符号化の深さ第１の性質が挙げられる。さらに、サブバンドを符号化する場合、本発明の有利な側面には各々のサブバンドを別々に階層符号化することが含まれる。

【００８５】（好適な実施の形態）本好適な実施の形態はウェイブレット変換処理を利用したもので、画像データウェイブレット変換手段による処理をまず最初に行なうものである。ウェイブレット変換処理の説明は多くの標準的なテキストに記載があり、特に、前述したStollnitzらによる本に記載がある。ここで、標準的なウェイブレット処理の概要を添付図面を参照して説明する。

【００８６】まず、図１３を参照する。オリジナル画像１は離散ウェイブレット変換（ＤＷＴ）を利用して４つのサブ画像３～６に変換される。サブ画像あるいはサブバンドは通常ＬＬ１、ＨＬ１、ＬＨ１、およびＨＨ１で示される。サブバンド名の最後についている１は分解レベルが１であることを示す。ＬＬ１サブバンドはオリジナル画像のローパス縮小版となる。

【００８７】利用されたウェイブレット変換は、例えば、Ｈａａｒ基礎機能、Ｄａｕｂｅｃｈｉｅｓ基礎機能などを変化させることができ、また、包含することもできる。次に、今度はＬＬ１サブバンドが利用されて、図１４に示されるように第２の離散ウェイブレット変換が適用され、サブバンドＬＬ２（８）、ＨＬ２（９）、ＬＨ２（１０）、ＨＨ２（１１）が得られる。このような処理が繰り返され、図１５に示すようになる。このＬＬ４バンド分解処理は、ＬＬ４サブバンドを伴ったオクターブバンドフィルタバンクと呼ばれ、さらに、ＬＬ４サブバンドはＤＣサブバンドと呼ばれる。当然、さらにいかなるレベルまで分解できるかは入力画像のサイズに依存する。

【００８８】元画像を得るためには、各レベルのＤＷＴを順番に逆に行なえばよい。このように、あるＪレベルのＤＷＴは逆に行なうこともでき、Ｊ単一レベルの逆ＤＷＴと呼ぶ。

【００８９】階層的に画像を符号化する場合、ＤＣサブバンドを符号化することで処理を開始することができる。そうして、残りのサブバンドは、レベルが減少するように順番に符号化する。４レベルＤＷＴについていえば、レベル４のサブバンド、つまり、ＨＬ４、ＬＨ４、そしてＨＨ４サブバンドはＤＣサブバンド（ＬＬ４）の後に符号化される。次に、レベル３（ＨＬ３、ＬＨ３、およびＨＨ３）のサブバンドが符号化され、続いてレベル２（ＨＬ２、ＬＨ２、およびＨＨ２）が、それからレベル１（ＨＬ１、ＬＨ１、およびＨＨ１）が符号化される。

【００９０】標準の画像に関していえば、符号化されたサブバンドは通常、画像中の「細々とした」情報を含む。それゆえ、それらはしばしば値のまばらなビット列を含み、実質的な圧縮は、サブバンドを量子化し、それらサブバンドのまばらなマトリクス形態を効率的に符号化することにより達成できる。

【0091】この符号化処理の概要は図16のフローチャート20に示されている。この符号化処理は画像データに離散ウェイブレット変換を施し、通常のサブバンドを生成することにより開始される（ステップ22）。次に、それらのサブバンドは最低周波数のものから最高周波数のものに向けて階層順に連結される（ステップ23）。

【0092】各サブバンドで3つのステップ24～26が実行される。ステップ24では、そのサブバンドの、最適なクワッドツリーと、それに応じた量子化ファクタを決定する。次に、各領域は適当な量子化手段でクワッドツリーによって特定されるように量子化される（ステップ25）。次に量子化されたサブバンドの値と適当なクワッドツリー情報が符号化される（ステップ26）。

【0093】図17において、復号処理が示されている。各サブバンドは連結された通りの順番に扱われ（ステップ31）、逆処理32～34はそのサブバンドに適用される。最初のステップ32では量子化及びクワッドツリー情報が復号される。次に、量子化された係数がステップ34で逆量子化処理される前に、復号される。求まったデータは、オリジナルデータを生成すべく逆離散ウェイブレット変換を施される（ステップ35）本好適な実施の形態の核心部分は最適なクワッドツリー及びそれに関連する量子化ファクタをサブバンドについて決定する処理（ステップ24）にある。各サブバンドはクワッドツリー構造を用いて様々なサイズの領域に区切られる。そしてその各領域に関連してクワッドツリーにおけるリーフが量子化のファクタとなる。クワッドツリーの利用は標準的なラインに沿って行なわれる。クワッドツリー及びそれに変えて適用できるデータ構造についての全体説明として、1984年6月版コンピュータサーベイ187頁～260頁にあるHanan Samet著の「クワッドツリーと関連階層データ構造」と題した調査文献が参考とできる。本好適な実施の形態では、各領域は、インデックス値として表現される量子化パラメータを用いて画一的に量子化される。この各サブバンドのインデックス値は、そのクワッドツリー構造、及び、そのクワッドツリーの各リーフに関する量子化データと共に符号化される。

【0094】多くの様々な符号化処理が利用できるが、本実施の形態では、その量子化データの符号化は、SWEET原理を利用して達成することができる。これは、上述のオーストラリア仮特許出願No．PO4728において開示されたものであって、以下ではSWEET符号化を称する。これから明らかにするが、本発明は、ダイナミック量子化を利用するものの、従来のSWEET符号化とは異なるものである。

【0095】一つのサブバンド内で空間的に量子化を変化できるように、クワッドツリー構造を用いてイメージを分割する。クワッドツリーは一つの領域からいくつかの様々な大きさのサブイメージへの分割を表現するのに用いられる。例えば、方形の領域或いは係数の配列を与えられると、その領域を標準的クワッドツリー技術に従って、4つのサブ領域に分割することができる。それぞの領域内で、所定の基準に従い、繰り返し分割することも可能であり、或いはその領域を「そのまま」にしておくことも可能である。

【0096】例えば、図18がクワッドツリー構造の例を示している。この図の構造によれば、難なくクワッドツリーを理解することができるであろう。この点において、領域40は4つのサブ領域に分割され、領域41及び42のみが「サブ分割」されている。右下の領域42は更にサブ分割されて領域43を形成している。

【0097】クワッドツリーの各リーフノードについて、量子化パラメータが決定される。異なる量子化ファクタは各サブ領域に提供される。クワッドツリー40の構造及び関連量子化パラメータは圧縮画像ビットストリームの中に符号化される。復号処理においては、各サブ領域を適当な量子化ファクタで逆量子化するためにこの情報が用いられる。

【0098】クワッドツリーは二進分割デシジョンの一つのシーケンスと表現することもできる。「1」は、領域の1分割を表すために用いられる。一方、「0」は一つのリーフノードを表すため、或いはその領域が分割されていないことを表すために用いられる。

【0099】表現に深さ優先のアプローチを用いると、1度の領域分割を示す1というビットのすぐ後には、対応するサブ領域の2進方で表した分割デシジョンがある。

【0100】サブ領域の分割デシジョンは、左上、右上、左下、右下の領域の順に符号化できる。このようにすると、図18のクワッドツリー40は11000000101000000で示される。ここで、最初の「1」は全（オリジナル）領域40を4つのサブ領域に分割することを示す。次の「1」はオリジナル領域の左上の四分の一領域41を分割することを示している。次の4つの「0」は、領域41のこれらのサブ領域がリーフノードであることを示している。最後に残ったビット10100000はオリジナル領域の右下の四分の一領域42の分割構造を表現している。

【0101】クワッドツリーの各リーフノードには関連する量子化手段が用いられる。クワッドツリーにおいてリーフノードが発生する順番に各リーフノードについての量子化パラメータを単純に符号化することも可能である。そのようなクワッドツリー表現を復号するにあたり、復号手段はその順番を計算して、各量子化パラメータを順番に復号する。

【0102】この好適な量子化処理には、デッドゾーンに対し画一的な量子化を行なうことが含まれる。離散ウェイブレット変換係数は整数値に量子化される。係数値

をcで、量子化された値をdで、その係数が存在する領域の量子化パラメータをqで表すと、ここで言う量子化は以下のように定義される。

【0103】

【数6】

$$d = fix\left(\frac{c}{q}\right)$$

ここで、fixは以下のように定義される。

【0104】

【数7】

$$fix(x) = \begin{cases} \lfloor x \rfloor & x \geq 0 \\ \lceil x \rceil & x \leq 0 \end{cases}$$

そして、└ ┘は通常「フロア」と呼ばれ、その数値以下の最も近い整数に丸める演算子である。また、┌ ┐は、通常「ルーフ」と呼ばれ、その数値以上の最も近い整数値に丸める演算子である。量子化パラメータqは、量子化ファクタである。符号化手段において、サブバンドの各係数は、この数式を用いて一つの整数値に量子化される。

【0105】逆量子化は、以下のように与えられる。

【0106】

【数8】

$$\hat{c} = q \times d + sign(d) \times \frac{q}{2}$$

ここで、

【0107】

【数9】

$$sign(d) = \begin{cases} -1 & d < 0 \\ 0 & d = 0 \\ 1 & d > 0 \end{cases}$$

＊である。

【0108】復号手段において、各係数は、この逆量子化式を用いて逆量子化される。

【0109】与えられた圧縮率、例えばRビット／ピクセル（bpp）という圧縮率に対して、最適量子化クワッドツリーと、それに関する量子化ファクタは、最小ゆがみを持つ圧縮画像を結果として導くように、クワッドツリーとして定義される。つまり、Rbpp或いは更に小さく画像を圧縮する際に、他のクワッドツリーを用いれば、より大きなゆがみを持つことになる。取り得る量子化ファクタのセットは、固定有限のセットであると推定される。本実施の形態では、1から16までの16の量子化ファクタを取り得る。16の異なる量子化ファクタに関し、シンプルコードは、与えられたファクタを決定するのに4ビット必要となる。

【0110】最適クワッドツリーを見つける方法は制約つき最適化のためのラグランジュ乗数アプローチを用いることにある。つまり、与えられたλ＞0に対し、

min［cost＝d（Q）＋λb（Q）］（数式1）

を求める。この最小化は、全てのクワッドツリー及び関連する量子化ファクタに対して行なう。クワッドツリー及び関連する量子化ファクタは数式1の中で、Qで表されている。これにより、この最小化は取り得るQの値全てについて行なわれる。もし、非制約ラグランジュプロブレムに対する最適な解が、Q＊で与えられれば、このQ＊は、その場合の制約プロブレムに対する最適な解になる。ここで、与えられた圧縮率はb（Q＊）である。

【0111】与えられた圧縮率に対する最適解を得るため、b（Q＊）が求める圧縮率Rに十分に近づくまで、分割法を用いてλを単純変化させることができる。

【0112】与えられた画像領域に対し数式1の最小化を行なうために用いる方法を、以下の擬似コードに示す。

```
*
[bits, distortion, nmax] = findQuantisationQuadtree (region, n, λ)
{
*/
**領域の最適固定量子化ファクタを見つける
*/
[bits, distortion, q] = findOptimumFixedQuantisation (region, n, λ);
set fixedQuantisationCost = distortion + λbits;
/*
**領域のクワッドツリー可変量子化を用いた場合のコストを見つける
**qmin は最小量子化ファクタである
*/
set nmax = largest bit number in region quantised with qmin;
partition region into 4 subregions;
```

```
   25
 [bits1, distortion], nmax1] = findQuantisationQuadtree region1, nmax,
λ);
 [bits2, distortion2, nmax2] = findQuantisationQuadtree region2, nmax,
λ);
 [bits3, distortion3, nmax3] = findQuantisationQuadtree region3, nmax,
λ);
 (bits4, distortion4, nmax4] = findQuantisationQuadtree region4, nmax,
λ);
 set bitSavings = 3 × (nmax - max(nmax1, nmax2, nmax3, nmax4);
 set variableQuantisationCost = distortion1 + distortion2 + distortion3
+ distortion4 +λ(bits1 + bits2 + bits3 + bits4 + (n-nmax+1) - bitSaving
s + 1);
 /*
 **可変量子化と固定量子化のうち、より適当なアプローチを見つける.
 */
 if variableQuantisationCost < fixedQuantisationCost
 {
  distortion = distortion1 + distortion2 + distortion3 + distortion4;
  bits = bits1 + bits2 + bits3 + bits4 + (n-nmax+1) - bitSavings + 1;
  nmax = max(nmax1, nmax2, nmax3, nmax4);
 }
else // 固定量子化が望ましい(リターンパラメータは既に正しく設定されてい
る)
 {
  output region and quantisation factor q
 }
}
 [bits, distortion, q] = findOptimumFixedQuantisation (region, n,λ
)
{
 /*
 **最小ラグラジアンコストに関する量子化ファクタを見つける
 */
 set cost = infinity;
 loop through the quantisation factors qi
 {
  quantise region with quantisation factor qi;
  bitsQ = SWEET code (region, n);
  set distortionQ to the quantisation distortion;
  if distortionQ +λbitsQ < cost
  {
   cost = distortionQ +λbitsQ
   bits = bitsQ;
   distortion = distortionQ;
   q = qi;
  }
 }
 /*
 **固定量子化ファクタを符号化するのに必要なビット数と
 **量子化クワッドツリーのための最終ビットを代入する
```

```
*/
set bits = bits + numQuantisationFactorBits + 1;
}
```

このコードには、16個の取り得る量子化パラメータのセットが内在する。

【0113】上記 findOptimumQuadtreeファンクションでは、全ての領域に対して用いられる最適固定量子化ファクタと、それに関するコストがいわゆる findOptimumFixedQuantisation ファンクションとして計算される。そして、その領域においてクワッドツリー可変量子化を用いた場合のコストが計算される。これは、その領域を4つのサブ領域に分割し、各サブ領域で繰り返しfindOptimumQuadtreeファンクションを呼び出すことにより行なわれる。そして、それらの固定量子化コスト及び可変量子化コストが比較され、そのコストが少ないほうが選択される。固定量子化が選択された場合には、その領域及び関連する量子化ファクタが出力される。この出力は後に、クワッドツリー及び関連する量子化ファクタを符号化し、符号化前の画像領域を量子化するために用いられる。

【0114】FindOptimumQuadtreeファンクションのvariableQuantisationCostの計算におけるビット数ファクタ「$(n-n_{max}+1)$ - bitSavings 」は、4つのサブ領域を符号化するためにSWEET符号化方法において必要となるエクストラビットの数であり、ビット番号パラメータnで表される。variableQuantisationCostの計算におけるビット数ファクタに付加される最後のエクストラビットは、量子化クワッドツリーにおいて領域の分割を示すのに用いられた「1」というビットである。

【0115】findOptimumFixedQuantisation において、SWEET符号化方法は、量子化された領域を符号化するために用いられるビット数の計算に用いられる。最適量子化ファクタとして、ラグラジアンコスト「distortion + λbits」を最小化するものが導かれる。最後に、その領域を符号化するための全ビット数に対し、numQuantisationFactorBitsと、1つのビットが加えられる。このnumQuantisationFactorBits は、単に、量子化ファクタ（本実施の形態では、これは4ビットである）を符号化する為に必要なビット数であり、また、前記1つのビットは、その領域がリーフノードであることを示すクワッドツリー最終ビット0である。

【0116】（装置の望ましい実施の形態）本発明の実施の形態は例えば図8に図示してあるような従来の汎用コンピュータを用いて実施するのが望ましく、図16または図17の処理が、コンピュータ上で動作するソフトウェアとして実行される。

【0117】詳しくは、符号化及び／又は復号の手順はコンピュータで実行されるソフトウェア内の指示によって影響される。ソフトウェアは2つの部分に分割できる。つまり、符号化及び／又は復号方法を実行するため＊の1つの部分と、ユーザーとの間のユーザーインタフェースを管理する別の部分、である。

【0118】そのソフトウェアは、例として以下に説明されている記憶デバイスを含むコンピュータで読みとり可能な媒体に記録されることができる。そのソフトウェアは、そのコンピュータで読みとり可能な媒体から、そのコンピュータへロードされ、そのコンピュータにより動作される。そのようなソフトウェアやコンピュータ・プログラムが記録されているコンピュータ読取り可能媒体は、コンピュータ・プログラム製品である。そのコンピュータ内でそのようなコンピュータ・プログラム製品を使用することは、本発明の実施の形態に関連してデジタル画像の符号化及びデジタル画像の符号化及び／又は復号に有利な装置にはむしろ効果的である。 コンピュータシステム800は、コンピュータ802とビデオディスプレイ816と、入力デバイス818と、820を含む。更に、ライン印刷装置、レーザ印刷装置、プロッタ、及びコンピュータ802に接続されている他の複写装置等を含む数種類の出力デバイスのいずれかをコンピュータシステム800に接続することができる。コンピュータシステム800は、モデム通信経路、コンピュータ・ネットワーク等のような適切な通信チャンネルを使用して通信インタフェースを介して、1台または2台以上の他のコンピュータに接続することができる。そのコンピュータ・ネットワークはローカル・エリア・ネットワーク（LAN）及び／又はワイド・エリア・ネットワーク（WAN）及び／又はイントラネット及び／又はインターネットを含むことができる。

【0119】コンピュータ802自体は中央演算処理ユニット（以下プロセッサ）804と、ランダム・アクセス・メモリ（RAM）とリード・オンリー・メモリー（ROM）と、入出力（I／O）インタフェース808と、ビデオインタフェース810と、1台又は2台以上の基本的に図8においてブロック8によって示されている記憶デバイスを有する。

【0120】記憶デバイス8には、フロッピーディスク、ハードディスク・ドライブ、光磁気ディスク・ドライブ、CD－ROM、磁気テープ、又は他多数の当業者に周知の不揮発性記憶デバイスの内1又は2以上のものを有することができる。

【0121】各成分804から812は典型的に、更にデータ、アドレス、制御バスを含むことのできるバス814経由で他の1台又は2台以上のデバイスに接続されている。

【0122】ビデオ・インタフェース810は、ビデオ・ディスプレイ816に接続されて、ビデオ・ディスプレイ816上でディスプレイするためコンピュータ80

2からビデオ信号を提供する。コンピュータ８０２を操作するためのユーザ入力は１台又は２台以上の入力デバイスにより提供されることができる。例えば、操作をする者は、キーボード８１８及び／又はコンピュータ８０２に入力を提供するためのマウス８２０のようなポインティング・デバイスを使用できる。

【０１２３】そのコンピュータ・システム８００は図示の目的で単に提供されたものであり、本発明の趣旨および範囲から逸脱することなく他の設定を使用することができる。コンピュータ・システムはＩＢＭ－ＰＣ／ＡＴまたはその互換機、またはマッキントッシュ（ＴＭ）ファミリーのＰＣやサン・マイクロシステムズのSparcstation（ＴＭ）またはその他のうちのひとつを用いて実現することができる。前述のコンピュータの種類は本発明の実施の形態を実施することのできる一例に過ぎない。概して、以下示される実施の形態における処理は、ソフトウェア、又は、コンピュータで読みとり可能な媒体としてのハードディスクドライブ（基本的に図８でブロック８１２として描写されている）上に記録されたプログラム、として内在し、プロセッサ８０４を用いて読みとり、制御される。プログラムと画素のデータと、ネットワークから取ってきたデータのいずれもの中間記録は、ハードディスクドライブ８１２と協調している可能性のあるセミコンダクタ・メモリ８０６を用いて行うことも可能である。

【０１２４】幾つかの場合においては、そのプログラムはＣＤ－ＲＯＭ又はフロッピーディスク（共に基本的にブロック８１２により描写される）上で復号され、ユーザに提供されることができ、あるいは、例えば、代りにユーザが、コンピュータに接続したモデム・デバイス経由でネットワークから読みとることもできる。

【０１２５】更には、そのソフトウェアは磁気テープ、ＲＯＭ又は集積回路、光磁気ディスク、コンピュータと他のデバイスの間の電磁波又は赤外線伝達チャンネル、ＰＣＭＣＩＡカードのようなコンピュータで読みとり可能なカード、Ｅ－ｍａｉｌ通信やウェブサイトに記録された情報を含むインターネットやイントラネット等、を含む他のコンピュータで読みとり可能な媒体から、コンピュータシステム８００にロードすることもできる。前述のものは適切なコンピュータで読みとり可能な媒体の一例にすぎない。本発明の趣旨及び範囲から逸脱することなくコンピュータで読みとり可能な他の媒体を使用することができる。

【０１２６】上記符号化及び／又は復号の方法は、符号化及び復号をその機能として或いはサブ機能として実行する１つ以上の集積回路などの専用ハードウェアで実行してもよい。そのような専用ハードウェアは、グラフィックプロセッサ、デジタル信号プロセッサ、または１つ以上のマイクロプロセッサ及びメモリを含んでもよい。

【０１２７】前記の内容は本発明の実施の形態のわずかな部分を説明したにすぎず、いかなる修正、及び／又は、変更も、本発明の範囲と趣旨を逸脱しない限りにおいて、この技術分野における当業者によってなされ得るものである。それゆえに、本実施の形態は図示されている全ての点を考慮するもので、制限的ではない。

【図面の簡単な説明】

【図１】図１は、本特許出願で説明されている画像表現技術を示す高レベルブロック図である。

【図２】図２は、本特許出願で説明されている分割を示す図である。

【図３】図３は、本特許出願で説明されている画像の表現または符号化の方法を示すフローチャートである。

【図４】図４は、図３での領域の符号化ステップを示す詳細なフローチャートである。

【図５】図５は、図３の方法に従って作成した画像の符号化表現の復号方法を示すフローチャートである。

【図６】図６は、図５での領域の復号ステップを示す詳細なフローチャートである。

【図７】図７（ａ）～（ｄ）は、図３から６の符号化および復号の方法に従った２次元、８係数領域の処理を示す図である。

【図８】図８は、汎用目的のコンピュータのブロック図である。

【図９】図９は、本特許出願で説明された別の方法による画像表現または符号化を示すフローチャートである。

【図１０】図１０は、本特許出願で説明された別の方法による画像表現または符号化を示すフローチャートである。

【図１１】図１１は、本特許出願で説明された別の方法による画像表現または符号化を示すフローチャートである。

【図１２】図１２は、本特許出願で説明された別の方法による画像表現または符号化を示すフローチャートである。

【図１３】図１３は、画像データに対するウェイブレット変換処理を説明する図である。

【図１４】図１４は、画像データに対するウェイブレット変換処理を説明する図である。

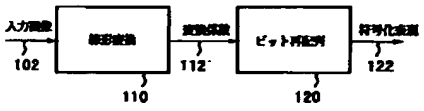
【図１５】図１５は、画像データに対するウェイブレット変換処理を説明する図である。

【図１６】図１６は符号化処理のフローチャートである。

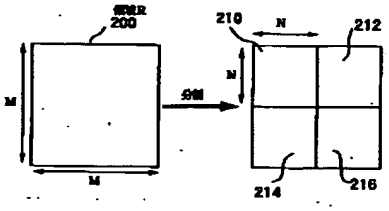
【図１７】図１７は復号処理のフローチャートである。

【図１８】図１８は、クワッドツリー分割の一例を示す図である。

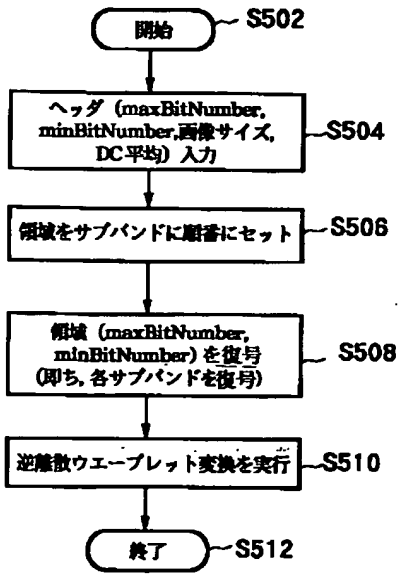
【図1】


102
110
112
120
122

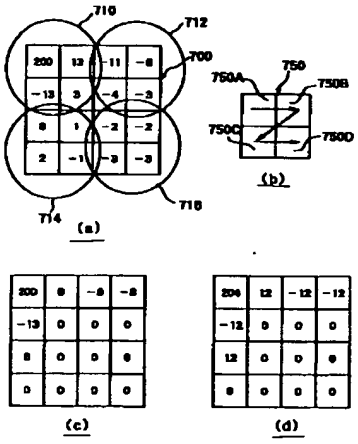
【図2】


200
210
N
212
M
M
M
214
216

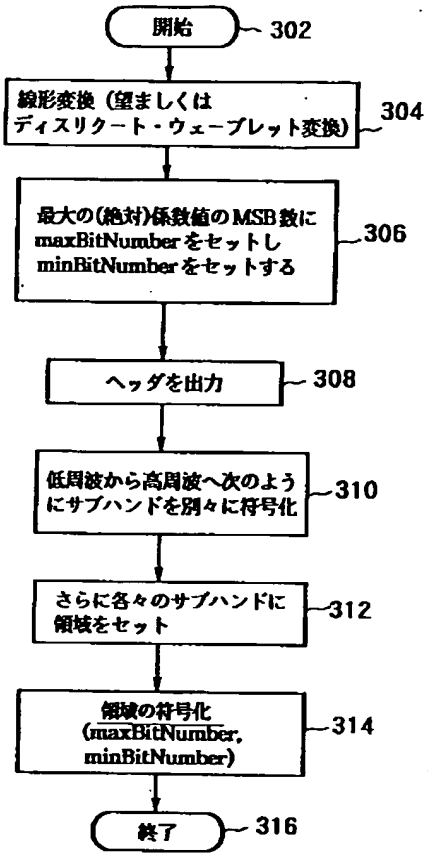
【図5】


開始
S502
ヘッダ（maxBitNumber, minBitNumber,画像サイズ, DC平均）入力
S504
領域をサブバンドに順番にセット
S506
領域（maxBitNumber, minBitNumber）を復号（即ち，各サブバンドを復号）
S508
逆離散ウエーブレット変換を実行
S510
終了
S512

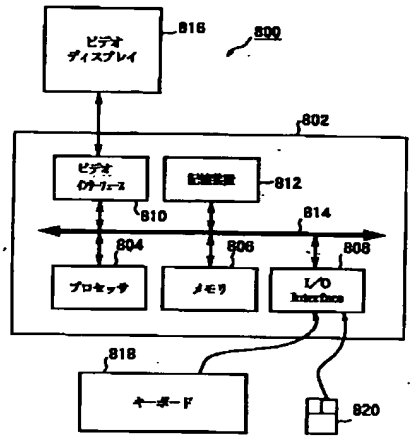
【図7】


710
712
700
714
716
750
750A
750B
750C
750D
(a)
(b)
(c)
(d)

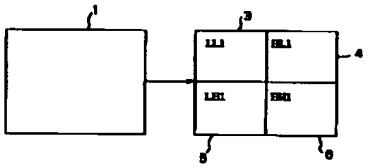
【図3】


開始
302
線形変換（望ましくは
ディスリクート・ウェーブレット変換）
304
最大の（絶対）係数値のMSB数に
maxBitNumberをセットし
minBitNumberをセットする
306
ヘッダを出力
308
低周波から高周波へ次のよう
にサブバンドを別々に符号化
310
さらに各々のサブバンドに
領域をセット
312
領域の符号化
(maxBitNumber,
minBitNumber)
314
終了
316

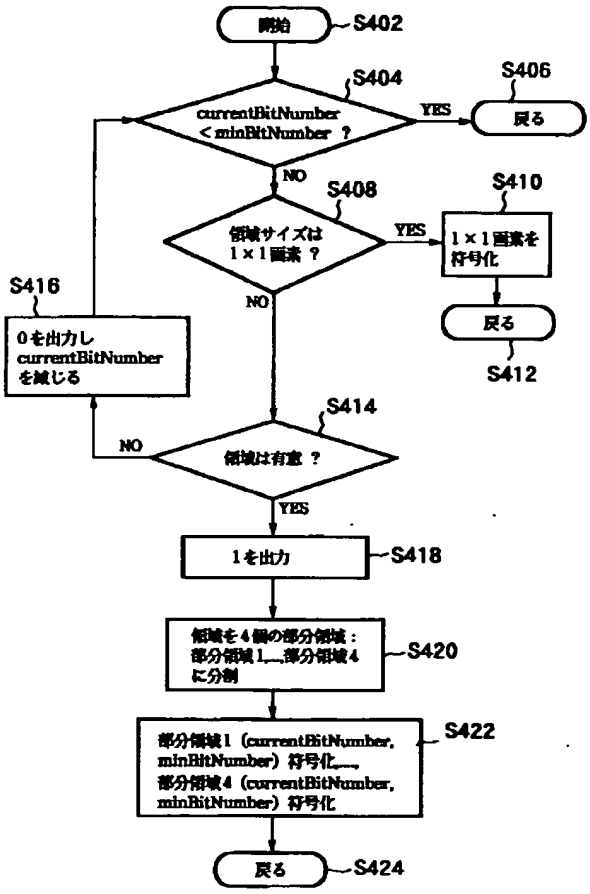
【図8】


ビデオ
ディスプレイ
816
800
802
ビデオ
インターフェース
812
810
814
804
806
808
プロセッサ
メモリ
I/O
Interface
818
キーボード
820

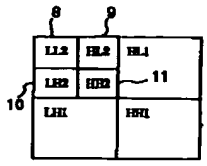
【図13】


1
3
LL1
HL1
LH1
HH1
4
5
6

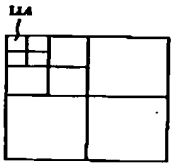
【図4】


開始
S402
S404
currentBitNumber
＜minBitNumber ？
YES
S406
戻る
NO
S408
領域サイズは
1×1画素 ？
YES
S410
1×1画素を
符号化
戻る
S412
NO
S416
0を出力し
currentBitNumber
を減じる
S414
領域は有意 ？
NO
YES
1を出力
S418
領域を4個の部分領域：
部分領域1,...部分領域4
に分割
S420
部分領域1（currentBitNumber,
minBitNumber）符号化,...,
部分領域4（currentBitNumber,
minBitNumber）符号化
S422
戻る
S424

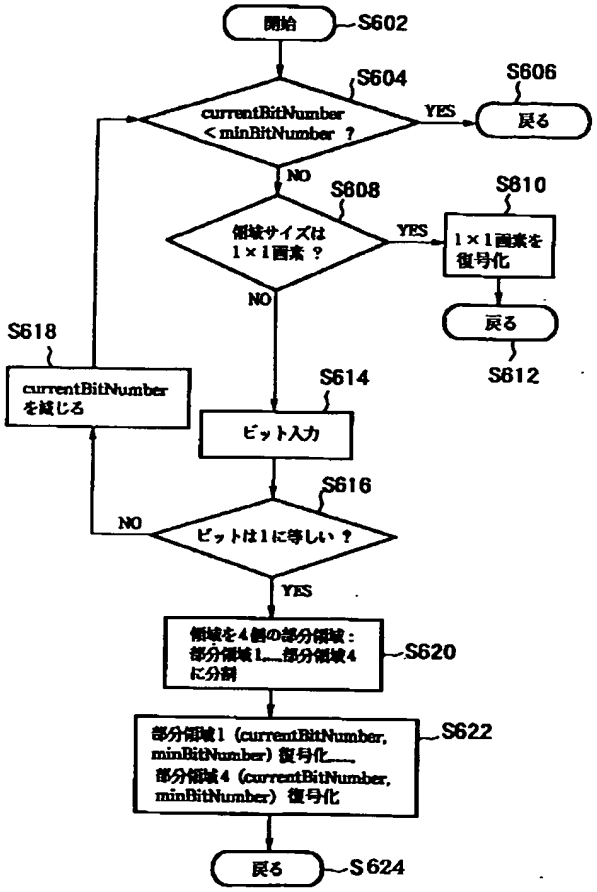
【図14】


8
9
LL2
HL2
HL1
LH2
HH2
11
10
LH1
HH1

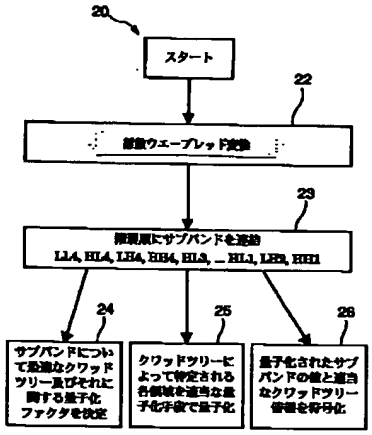
【図15】


LL4

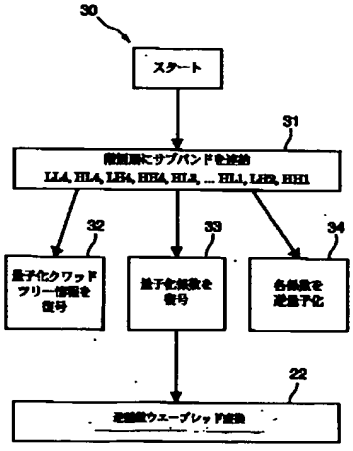
【図6】


開始
S602
S604
currentBitNumber
< minBitNumber ？
YES
S606
戻る
NO
S608
領域サイズは
1×1画素 ？
YES
S610
1×1画素を
復号化
戻る
S612
NO
S618
currentBitNumber
を減じる
S614
ビット入力
S616
NO
ビットは1に等しい ？
YES
領域を4個の部分領域：
部分領域1,...部分領域4
に分割
S620
部分領域1（currentBitNumber,
minBitNumber）復号化,...
部分領域4（currentBitNumber,
minBitNumber）復号化
S622
戻る
S624

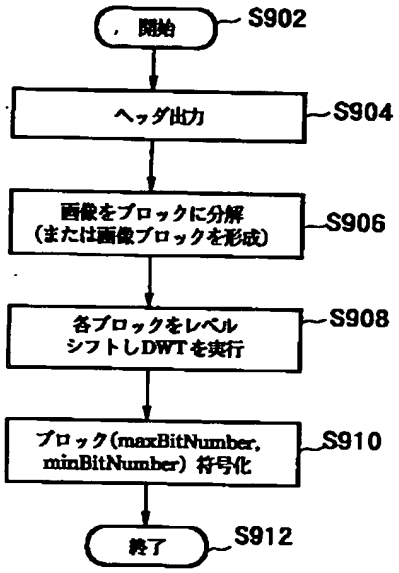
【図16】


20
スタート
22
逆量子化ウェーブレット変換
23
階層順にサブバンドを連結
LL4, HL4, LH4, HH4, HL3, ... HL1, LH1, HH1
24
25
26
サブバンドについて最適なクワッドツリー及びそれに関する量子化ファクタを決定
クワッドツリーによって特定される各領域を適当な量子化手法で量子化
量子化されたサブバンドの値と適当なクワッドツリー情報を符号化

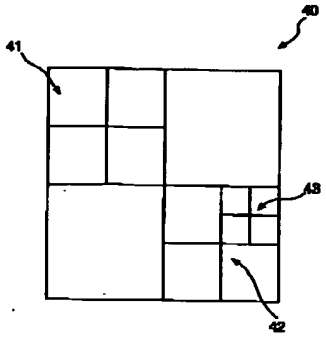
【図17】


30
スタート
31
階層順にサブバンドを連結
LL4, HL4, LH4, HH4, HL3, ... HL1, LH1, HH1
32
33
34
量子化クワッドツリー情報を復号
量子化係数を復号
各係数を逆量子化
22
逆離散ウェーブレット変換

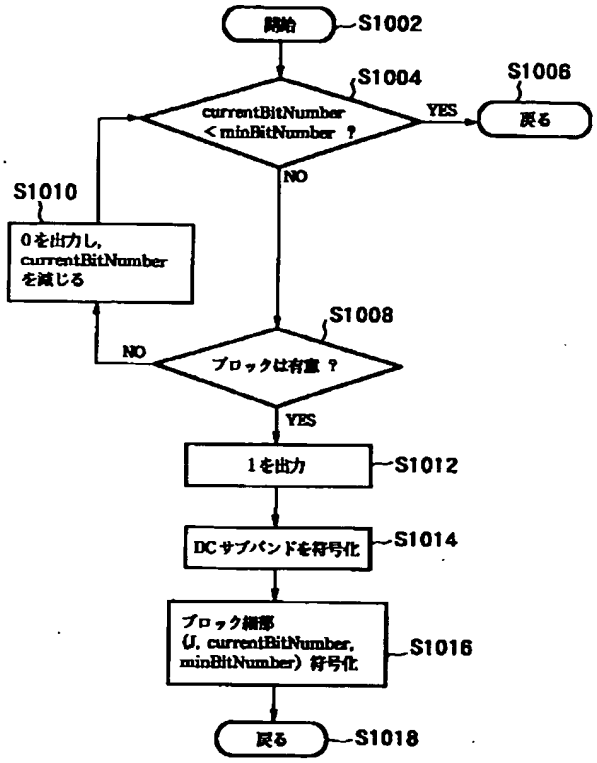
【図9】


開始
S902
ヘッダ出力
S904
画像をブロックに分解
（または画像ブロックを形成）
S906
各ブロックをレベル
シフトしDWTを実行
S908
ブロック(maxBitNumber,
minBitNumber) 符号化
S910
終了
S912

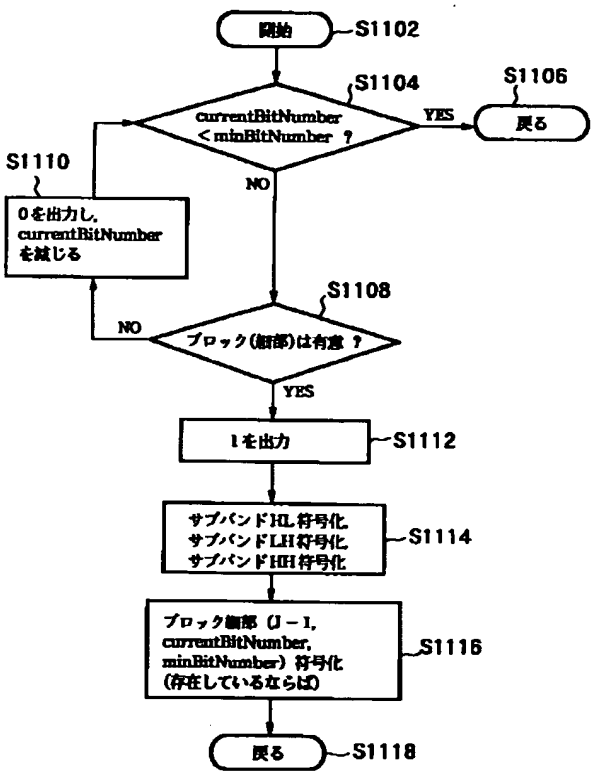
【図18】


40
41
43
42

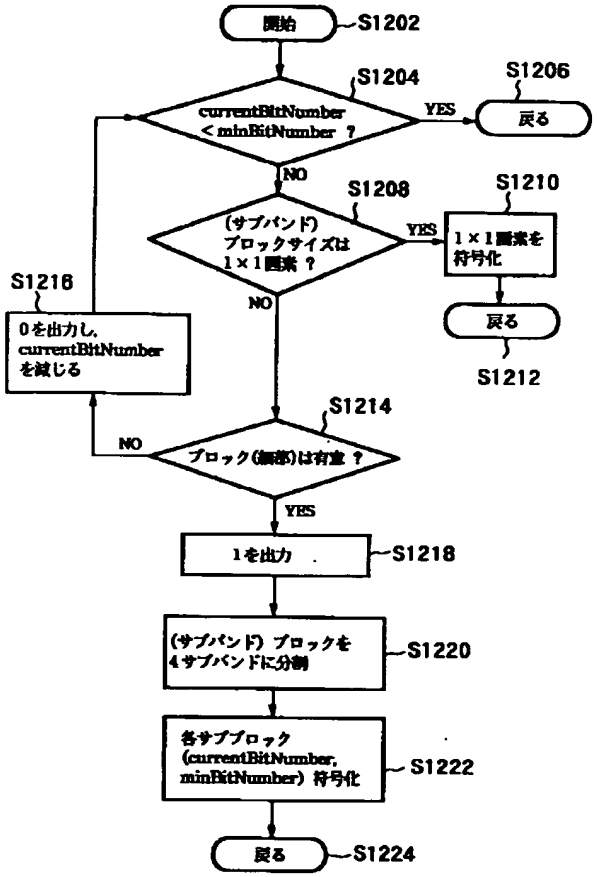
【図10】


開始
S1002
S1004
currentBitNumber
＜minBitNumber ？
YES
S1006
戻る
NO
S1010
0を出力し，
currentBitNumber
を減じる
S1008
NO
ブロックは有意 ？
YES
1を出力
S1012
DCサブバンドを符号化
S1014
ブロック細部
(J, currentBitNumber,
minBitNumber) 符号化
S1016
戻る
S1018

【図11】


開始
S1102
S1104
currentBitNumber
＜minBitNumber ？
YES
S1106
戻る
NO
S1110
0を出力し，
currentBitNumber
を減じる
S1108
NO
ブロック(領部)は有意 ？
YES
1を出力
S1112
サブバンドHL符号化
サブバンドLH符号化
サブバンドHH符号化
S1114
ブロック領部（J－1，
currentBitNumber，
minBitNumber）符号化
（存在しているならば）
S1116
戻る
S1118

【図12】



フロントページの続き

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　ジェイムズ　フィリップ　アンドリュー
オーストラリア国　2113　ニュー　サウス　ウェールズ州，ノース　ライド，トーマス　ホルト　ドライブ　1　キヤノン　インフォメーション　システムズ　リサーチ　オーストラリア　プロプライエタリー　リミテツド内

【外国語明細書】

1. TITLE OF THE INVENTION

A METHOD AND APPARATUS FOR DIGITAL DATA COMPRESSION

2. Claims

1. A method of compressing digital data including the steps of:

(a) transforming said data utilising a discrete wavelet transform to produce corresponding transformed data;

(b) quantising said transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of said quadtree leaf nodes has an associated quantisation factor utilised in said quantising of said transformed data.

2. A method as claimed in claim 1 wherein said quadtree is determined to be an optimum in a rate distortion sense.

3. A method as claimed in claim 1 wherein said quadtree is encoded utilising a binary prefix notation followed by a list of quantisation factors.

4. A method as claimed in claim 2 wherein the method of Lagrange multipliers is utilised to determine said optimum.

5. A method as claimed in claim 4 wherein said optimum is optimised for a predetermined bits per data item.

6. A method as claimed in claim 1,wherein said digital data includes image data.

7. A method as claimed in claim 1, wherein said digital data includes video data.

8. A method as claimed in claim 7 wherein said video data includes frame difference data.

9. A method of decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information, the method including the steps of

(a) decoding said quantization and quadtree information;

(b) decoding said encoded quantized coefficents

(c) inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and

(d) inverse transforming the inverse quantized coefficients.

10. A method as claimed in claim 9,wherein said digital data includes image data.

11. A method as claimed in claim 9, wherein said digital data includes video data.

12. A method as claimed in claim11, wherein said video data includes frame difference data.

13. An apparatus for compressing digital data including the apparatus including:

transformation means for transforming said data utilising a discrete wavelet transform to produce corresponding transformed data;

quantization means for quantizing said transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of said

quadtree leaf nodes has an associated quantisation factor utilised in said quantising of said transformed data.

14. An apparatus as claimed in claim 13 wherein said quadtree is determined to be an optimum in a rate distortion sense.

15. An apparatus as claimed in claim 13 wherein said quadtree is encoded utilising a binary prefix notation followed by a list of quantisation factors.

16. An apparatus as claimed in claim 15 wherein Lagrange multipliers is utilised to determine said optimum.

17. An apparatus as claimed in claim 16 wherein said optimum is optimised for a predetermined bits per data item.

18. An apparatus as claimed in claim 13, wherein said digital data includes image data.

19. An apparatus as claimed in claim 13, wherein said digital data includes video data.

20. An apparatus as claimed in claim 19, wherein said video data includes frame difference data.

21. An apparatus for decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information, the apparatus including;

means for decoding said quantization and quadtree information;

means for decoding said encoded quantized coefficents

means for inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and

means for inverse transforming the inverse quantized coefficients.

22. An apparatus as claimed in claim 21, wherein said digital data includes image data.

23. An apparatus as claimed in claim 21, wherein said digital data includes video data.

24. An apparatus as claimed in claim 23, wherein said video data includes frame difference data.

25. A computer program product including a computer readable medium having recorded thereon a computer program for compressing digital data, the computer program product including:

transformation means for transforming said data utilising a discrete wavelet transform to produce corresponding transformed data;

quantization means for quantizing said transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of said quadtree leaf nodes has an associated quantisation factor utilised in said quantising of said transformed data.

26. A computer program product as claimed in claim 25 wherein said quadtree is determined to be an optimum in a rate distortion sense.

27. A computer program product as claimed in claim 25 wherein said quadtree is encoded utilising a binary prefix notation followed by a list of quantisation factors.

28. A computer program product as claimed in claim 27 wherein Lagrange multipliers is utilised to determine said optimum.

29. A computer program product as claimed in claim 28 wherein said optimum is optimised for a predetermined bits per data item.

30. A computer program product as claimed in claim 25, wherein said digital data includes image data.

31. A computer program product as claimed in claim 25, wherein said digital data includes video data.

32. A computer program product as claimed in claim 31, wherein said video data includes frame difference data.

33. A computer program product including a computer readable medium having recorded thereon a computer program for decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information,, the computer program product including:

means for decoding said quantization and quadtree information;

means for decoding said encoded quantized coefficents

means for inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and

means for inverse transforming the inverse quantized coefficients.

34. A computer program product as claimed in claim 33, wherein said digital data includes image data.

35. A computer program product as claimed in claim 33, wherein said digital data includes video data.

36. A computer program product as claimed in claim 35, wherein said video data includes frame difference data.

3. Detailed Description of the Invention

Field of Invention

The present invention relates to the field of computer data compression with particular application to digital image compression. The present invention further provides for a spatially adaptive quantisation using quadtrees of discrete wavelet transform image and video data.

Background of Invention

The field of digital data compression and in particular digital image compression and digital video compression has attracted great interest for some time.

In the field of digital image compression, many different techniques have been utilised. In particular, one popular technique is the JPEG standard which utilises the discrete cosine transform to transform standard size blocks of an image into corresponding cosine components. In this respect, the higher frequency cosine components are heavily quantised so as to assist in obtaining substantial compression factors. The heavy quantisation is an example of a "lossy" technique of image compression. The JPEG standard also provides for the subsequent lossless compression of the transformed coefficients.

Recently, the field of wavelet transforms has gained great attention as an alternative form of data compression. The wavelet transform has been found to be highly suitable in representing data having discontinuities such as sharp edges. Such discontinuities are often present in image data or the like.

Although the preferred embodiments of the present invention will be described with reference to the compression of image data, it will be readily evident that the preferred embodiment is not limited thereto. For examples of the many different applications of Wavelet analysis to signals, reference is made to a survey article entitled "Wavelet Analysis" by Bruce et. al. appearing in IEEE Spectrum, October 1996 page 26 - 35. For a discussion of the different applications of wavelets in computer graphics, reference is made to "Wavelets for Computer Graphics", I. Stollinitz et. al. published 1996 by Morgan Kaufmann Publishers, Inc.

Further, digital compression techniques applied to video streams are also well known. For example, compression techniques relying on the video frame difference signals are also well known.

For objective and subjective reasons, better compression of digital signals can be obtained by varying the amount of quantisation of each region as compared to using a fixed amount of quantisation across a whole signal. Of course, usually some coding

overhead will be required to indicate how the quantisation has been varied across the signal.

In the past, adaptive quantisation has been employed, for example, in the video compression standard MPEG and its various implementations.

It would be desirable to provide for a more adaptive form of quantisation of a signal rather than the fixed techniques presently known.

Aspects of Invention

It is an object of the present invention to ameliorate one or more disadvantages of the prior art.

In accordance with a first aspect of the present invention, there is provided a method of compressing digital data including the steps of (a) transforming the data utilising a discrete wavelet transform to produce corresponding transformed data; (b) quantising the transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of the quadtree leaf nodes has an associated quantisation factor utilised in the quantising of the transformed data.

In accordance with a second aspect of the present invention there is provided a method of decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information, the method including the steps of: (a) decoding said quantization and quadtree information; (b) decoding said encoded quantized coefficents; (c) inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and (d) inverse transforming the inverse quantized coefficients.

In accordance with a third aspect of the present invention there is provided an apparatus for compressing digital data including the apparatus including: transformation means for transforming said data utilising a discrete wavelet transform to produce corresponding transformed data; quantization means for quantizing said transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of said quadtree leaf nodes has an associated quantisation factor utilised in said quantising of said transformed data.

In accordance with a fourth aspect of the present invention there is provided an apparatus for decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information, the apparatus including; means for decoding said quantization and quadtree information;
means for decoding said encoded quantized coefficents; means for inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree

information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and means for inverse transforming the inverse quantized coefficients.

In accordance with a fifth aspect of the present invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for compressing digital data, the computer program product including: transformation means for transforming said data utilising a discrete wavelet transform to produce corresponding transformed data; quantization means for quantizing said transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of said quadtree leaf nodes has an associated quantisation factor utilised in said quantising of said transformed data.

In accordance with a sixth aspect of the present invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for decompressing digital data, which digital data includes encoded quantized coefficients and associated quantization and quadtree information, the computer program product including: means for decoding said quantization and quadtree information; means for decoding said encoded quantized coefficents; means for inverse quantizing the decoded quantized coefficients in accordance with the quantization and quadtree information, wherein each quadtree leaf node has an associated quantization factor utilised in said inverse quantising of said decoded quantized coefficients; and means for inverse transforming the inverse quantized coefficients.

Detailed Description

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) and/or operation(s), unless the contrary intention appears.

Before proceeding with a description of the preferred embodiments, a description is given of the image compression and decompression method disclosed in Australian Provisional Patent Application No. PO 4728, entitled "A method for Digital Image Compression", filed on 22 January 1997 by Canon Information Systems Research Australia Pty. Ltd. This method of compression and decompression is described in the following sections hereinafter entitled *"1.0 Overview of SWEET Image Compression Method"*, *"1.1 Encoding Process of First SWEET Image Compression Method"*, *"1.2 Decoding Process of First SWEET Image Compression Method"*, *"1.3 Two-Dimensional Example"* , *and "1.4 Encoding Process of Second SWEET Image Compression Method"*.

1.0 Overview of SWEET Image Compression Method(s)

A high-level block diagram is illustrated in Fig. 1 to provide an overview of encoding method. An input image 102 is provided to the transform block 110, which is preferably a linear transform, to produce corresponding transform coefficients 112. A discrete wavelet transform (DWT) is preferably employed.

The two-dimensional DWT of an image is a transform that represents the image using a low frequency approximation to the image and three high frequency detail components. Conventionally, these components are termed subbands. Each of the four sub-images formed by the DWT is one quarter of the size of the original image. The low frequency image contains most of the information about the original

image. This information, or energy compaction, is the feature of the discrete wavelet transform image subbands that is exploited for image compression.

The single-level DWT can be applied recursively to the low frequency image, or subband, an arbitrary number of times. For example, a three-level DWT of the image is obtained by applying the transform once and then applying the DWT to the low subband resulting from the transformation. Thus, this results in 9 detail subbands and one (very) low frequency subband. Even after three levels of DWTs, the resulting low frequency subband still contains a significant amount of information of the original image, yet is 64 times smaller (1/4 x 1/4 x 1/4), thereby effecting a factor of 64 in compression.

However, other linear transformations for decorrelating image data may be practiced. For example, a discrete cosine transform (DCT) can be practiced. The transform coefficients 112, or more specifically the bit sequences representing their values, are then coded by the bit rearrangement block 120 in an efficient fashion to provide the coded representation 122.

The decoding process is simply the reverse of this encoding process. The encoded coefficients are decoded into the transform coefficients. The (transform domain) image is then inverse transformed to form the original image, or some approximation thereof.

Before proceeding with a further description of the embodiments, a brief review of terminology used hereinafter is provided. For a binary integer representation of a number, "bit n" or "bit number n" refers to the binary digit n places to the left of the least significant bit. For example, assuming an 8-bit binary representation, the decimal number 9 is represented as 00001001. In this number, bit 3 is equal to 1, while bits 2, 1, and 0 are equal to 0, 0, and 1, respectively. Furthermore, a transform may be represented as a matrix having coefficients arranged in rows and columns, with each coefficient represented by a bit sequence. Conceptually speaking the matrix may be regarded as having three dimensions; one dimension in the row direction; a second dimension in the column direction and a third dimension in the bit sequence direction. A plane in this three-dimensional space, which passes through each bit sequence at the same bitnumber, is called a bitplane or bit plane.

For transform coding applications, the number of bits per coefficient required to represent the possible range of coefficients is determined by the linear transform and the resolution of each pixel (in bits per pixel) in the input image. This range of values for each pixel is typically large relative to the values of most of the transform coefficients, and thus many coefficients have a large number of leading zeros. For example, the number 9 has four leading zeros in a 8-bit representation and has 12 leading zeros for a 16-bit representation. The compression method and apparatus

represents (or codes) these leading zeros, for blocks of coefficients, in an efficient manner. The remaining bits and sign of the number are encoded directly without modification.

To simplify and the description, the transform coefficients are assumed hereinafter to be represented in an unsigned binary integer form, with a single sign bit. That is, the decimal numbers -9 and 9 are represented with the same bit sequence, namely 1001, with the former having a sign bit equal to 1 to indicate a negative value, and the latter having a sign bit equal to 0 to indicate a positive value. The number of leading zeros is determined by the range of the transform coefficients. In using an integer representation, the coefficients are implicitly already quantised to the nearest integer value, although this is not necessary. Further, for the purpose of compression, any information contained in fractional bits is normally ignored.

A region includes a set of contiguous image coefficients. The term coefficient is used hereinafter interchangeably with pixel, however, as will be well understood by a person skilled in the art, the former is typically used to refer to pixels in a transform domain (eg., a DWT domain).

1.1 Encoding Process of First SWEET Image Compression Method

A more detailed description of the first image compression method is provided with reference to Figs. 3 and 4.

Fig. 3 is a flow diagram illustrating the first image encoding method. In step 302, processing commences using an input image. In step 304, the input image is transformed using a linear transformation, preferably a discrete wavelet transform. An initial region is defined to be the whole image. For example, in the case of a three-level DWT of the input image, the resulting coefficients consisting of the 10 subbands can be specified as the region. Alternatively each subband can be processed separately, setting each initial region to the whole subband in question.

In step 306, the most significant bit (msb) of the largest absolute value of the transform coefficients is determined and a parameter, maxBitNumber, is set to this coefficient value. For example, if the largest transform coefficient has a binary value of 00001001 (decimal 9), the parameter maxBitNumber is set to 3, since the msb is bit number 3. Alternatively, the parameter maxBitNumber may be set to be any value that is larger that the msb of the largest absolute value of the transform coefficients.

Further, in step 306, a coding parameter, minBitNumber is set to specify the coded image quality. In particular, this coding parameter specifies the precision of every coefficient in the transformed image and can be varied as required. For example, a minBitNumber of 3 provides a coarser reproduction of the original image than does a value of 1.

Optionally, the technique involves step 308, which provides an output header in the coded representation of the input image. Thus, in a practical implementation, header information is output as part of the coded representation. For example, the output header may contain information about the source image, including the image height and width, the number of levels of the DWT, the mean value of the DC subband, the maxBitNumber parameter, and the minBitNumber parameter.

Beginning in step 310, each subband of the transformed image is coded separately in steps 312 and 314. Each subband is coded independently, in order from low frequency to high frequency. For the DC subband, the mean value is removed prior to coding and coded into the header information in step 308. In step 312, each subband is coded by setting an initial region as the whole subband. In step 314, the region is encoded with the maxBitNumber and minBitNumber as parameters. This provides a hierarchical code, since lower resolution versions of the image are coded into the bit stream before higher resolutions. Processing terminates in step 316.

Fig. 4 is a detailed flow diagram of the procedure "Code region(currentBitNumber, minBitNumber)" called in step 314 of Fig. 3 for coding each region, where maxBitNumber is provided as the currentBitNumber. In step 402, processing commences. The inputs to the region coding process of Fig. 4 include the currentBitNumber and minBitNumber parameters. Preferably, the method is implemented as a recursive technique where the process is able to call itself with a selected region or sub-region. However, the process may implemented in a non-recursive manner.

In decision block 404, a check is made to determine if the currentBitNumber parameter is less than the minBitNumber parameter. Otherwise, if decision block 404 returns true (yes), nothing is done and processing returns to the calling procedure in step 406. This condition indicates that every coefficient in the selected region has a msb number less than minBitNumber. If decision block 404 returns false (no), processing continues at decision block 408.

In decision block 408, a check is made to determine if the selected region is a 1 x 1 pixel. If decision block 408 returns true (yes), processing continues at step 410. In step 410, the 1 x 1 pixel is coded. Preferably, this involves directly outputting the remaining bits above the minBitNumber in the coded representation. In step 412, processing returns to the calling procedure. Otherwise, if decision block 408 returns false (no), the region consists of more than one coefficient and processing continues at decision block 414.

In decision block 414, the selected region is checked to determine if it is significant. That is, the significance of the region is tested. The region is said to be insignificant if the msb number of each coefficient in the region is less than the value of

the currentBitNumber parameter. To make the concept of region significance precise, a mathematical definition is given in Equation (1). At a given bit number, say currentBitNumber = n, the region is said to be insignificant if:

$$|c_{ij}| < 2^n, \forall i,j \in R, \qquad (1)$$

where R denotes the region, and c_{ij} denotes coefficient (i,j) in this region.

If decision block 414 returns false (no), processing continues at step 416. In step 416, a value of 0 (or first token) is output in the coded representation stream, and the currentBitNumber parameter is decremented by 1. That is, the next, lower bitplane of the region is selected for processing. Processing then continues at decision block 404, where the region is again processed with the parameters currentBitNumber-1 and minBitNumber. Otherwise, if decision block 414 returns true (yes), that is, the region is significant, processing continues at step 418.

In step 418, a value of 1 (or second token) is output in the coded representation stream. In step 420, the selected region is partitioned into a predetermined number (preferably, 4) of subregions using a specified partitioning algorithm. The partitioning algorithm used is known to the decoder.

In this method, square regions are used. A region is partitioned preferably into 4 equal-sized (square) subregions. As shown in Fig. 2, the selected region (R) 200 has a size of M x M coefficients and is partitioned into four equal-sized subregions 210, 212, 214 and 216. Each of the subregions has a size of N x N, where N is equal to M/2. This is not always possible depending on the size and shape of the initial region. If this is not possible, the initial region can be partitioned into a number of square regions, each having dimensions that are a power of 2, and encode these partitions separately. In any case, this initialization has minimal effect on the overall results if done in an intelligent fashion. Alternatively, a different partition may be used that is suitable for a block-based coder.

In step 422, each subregion is then coded with the same currentBitNumber and minBitNumber parameters. This is preferably done by means of a recursive call to the procedure "Code region(currentBitNumber, minBitNumber)" of Fig. 4. This coding of subregions may be implemented in parallel or sequentially. In the latter case, the processing may commence from a low frequency subband to higher frequency subbands in turn.

In the coded representation, a transform coefficient is coded by simply outputting the pixel bits from the currentBitNumber to the minBitNumber. Preferably, a convention is followed whereby the sign is output only if some of the coefficient bits

were non-zero. For example, if currentBitNumber = 3, minBitNumber = 1, then -9 (00001001) is coded as "1 0 0" followed by a sign bit "1".

1.2 Decoding Process of First SWEET Image Compression Method

Fig. 5 is a flow diagram illustrating a method of decoding the coded representation of an image obtained using the process of Figs. 3 and 4. In step 502, processing commences using the coded representation. In step 504, the header information is read from the coded representation to determine the size of the original image, and hence the initial region size. Also, information such as maxBitNumber (equal to the initial currentBitNumber in the coding process) and minBitNumber are input. Further information includes the mean value of the DC subband.

In step 506, decoding of each subband is commenced by setting the region to the respective subbands in turn. In step 508, the selected region is decoded using the maxBitNumber and minBitNumber parameters. In step 510, the inverse DWT is applied to the decoded selected region. Processing terminates in step 512.

Fig. 6 is a detailed flow diagram of step 508 of Fig. 5 for decoding each region using procedure call "Decode region(currentBitNumber, minBitNumber)", where maxBitNumber is provided as the currentBitNumber. In step 602, processing commences. The inputs to the region decoding process of Fig. 6 are the currentBitNumber and minBitNumber parameters. Again, the method is preferably implemented as a recursive technique. However, the process can be implemented in a non-recursive manner.

In decision block 604, a check is made to determine if the currentBitNumber is less than the minBitNumber. If decision block 604 returns true (yes), processing continues at step 606, where processing returns to the calling procedure. Otherwise, if decision block 604 returns false (no), processing continues at decision block 608.

In decision block 608, a check is made to determine if the selected region has a size of 1 x 1 pixels. If decision block 608 returns true (yes), processing continues at step 610. In step 610, the 1 x 1 region is decoded. Processing then returns to the calling procedure in step 612. If decision block 608 returns false (no), processing continues at step 614. In step 614, a bit is input from the coded representation.

In decision block 616, a check is made to determine if the bit is equal to 1, that is, the input is checked to determine if the region is significant. If decision block 616 returns false (no), processing continues at step 618. In step 618, the currentBitNumber is decremented, and processing continues at decision block 604. Otherwise, if decision block 616 returns true (yes), processing continues at step 620. In step 620, the region is partitioned into the predetermined number (preferably, 4) of sub-regions. In step 622, each of the sub-regions is decoded using the currentBitNumber and

minBitNumber. This is carried out by means of a recursive call to the process illustrated in Fig. 6. In step 624, processing returns to the calling procedure.

Thus, the bits output from the significance decisions in the encoder instruct the decoder on which path of the algorithm to take, thus mimicking the encoder. The pixels, and possible sign, are decoded by simply reading in the appropriate number of bits (currentBitNumber to minBitNumber and if some of these are non-zero the sign bit).

1.3 Two-Dimensional Example

The method effectively codes the leading zeros of most transform coefficients, while coding the bits from the most significant bit to the predetermined least significant bit, specified by the parameter minBitNumber, and the sign simply as is. Thus, the compression method advantageously represents the leading zeros. This method is very efficient in certain situations, namely for coding discrete wavelet transform image coefficients, which typically exhibit a large dynamic range.
A few coefficients typically have very large values, while most have very small values.

An example of encoding a two-dimensional region including 4 x 4 coefficients is described with reference to Figs. 7A to 7D. The processing of the 4 x 4 region 700 of Fig. 7A is commenced with the maxBitNumber set to 7 since this is the largest bit number (bitplane) of all of the coefficients:

$$\begin{bmatrix} 200 & 13 & -11 & -8 \\ -13 & 3 & -4 & -3 \\ 8 & 1 & -2 & -2 \\ 2 & -1 & -3 & -3 \end{bmatrix}$$

The minBitNumber is set to 3, for illustrative purposes. A header is preferably output in the coded representation containing the maxBitNumber and minBitNumber. The process of coding the region 700 then follows.

At currentBitNumber = 7, a one (1) is output since the region 700 is significant with respect to bit number 7 (see decision block 404, 408, and 414 and step 418 of Fig. 4). The region 700 is then partitioned into four sub-regions (see step 420 of Fig. 4): the top left region 710, the top right region 712, the bottom left region 714 and the bottom right region 716 of Fig. 7A. Each of the subregions consist of 2 x 2 coefficients.

The sub-regions 710, 712, 714 and 716 of Fig. 7A are in turn coded in the predefined processing sequence shown of Fig. 7B, where a region 750 consists of four sub-regions 750A to 750D. The three arrows illustrated in the diagram indicate the

order or sequence of processing, that is, top left sub-region 750A, top right sub-region 750B, bottom left sub-region 750C, and bottom right sub-region-750D, respectively.

The sub-region 710 of Fig. 7A is coded first (see step 422 of Fig. 4). For the currentBitNumber equal to 7, a one (1) is output in the coded representation. The sub-region 710 is then partitioned into four 1 x 1 pixels having decimal values 200, 13, -13 and 3. Each of these coefficients is coded by outputting the bits of each coefficient from the currentBitNumber = 7 to the minBitNumber = 3 (see decision block 408 and step 410 of Fig. 4). A sign bit is then output if required. Thus, the decimal value is 200 is coded as 11001 followed by the sign bit 0. The coefficient value 13 is coded as 00001 with a sign bit 0. The coefficient value -13 is coded as 00001 with a sign bit 1. Finally, the coefficient value 3 is coded as 00000 (without a sign bit). The coded representation of each coefficient includes the two "1" bits preceding the bits of coefficients "200" between the currentBitNumber and minBitNumber. This completes the coding of the top left sub-region 710. The coded output at this state is:

$$\underbrace{1111001}_{200}\ \overset{\text{sign bit}}{\underbrace{0}}\ \underbrace{0000100000}_{13}\ \underbrace{1}_{-13}\ \underbrace{00000}_{3}.$$

The header information is not shown in the foregoing expression.

The top right sub-region 712 is then coded (per Fig. 7B). A zero (0) is output for each of currentBitNumber equal to 7, 6, 5, and 4, since the region 712 is insignificant with respect to these bit numbers. A one (1) is output at currentBitNumber = 3, since this bitplane is significant with respect to bit number 3. The sub-region 712 is partitioned into the four 1 x 1 pixels having values -11, -8, -4 and -3. These decimal values are coded as bit value 1 with sign bit 1, bit value 1 with sign bit 1 and bit values 0 and 0 without sign bits, respectively. Thus, at this stage, the coded representation is as follows:

$$11110010000010000011000000000111\ 1\ 1\ 0\ 0$$
$$-11\ -8\ -4\ -3$$

The bottom left sub-region 714 is then encoded. A zero (0) is output for each of currentBitNumber equal to 7, 6, 5, and 4, since the region 714 is insignificant with respect to these bit numbers. A one (1) is output at currentBitNumber equal to 3, since this bitplane is significant with respect to bit number 3. The sub-region 714 is then partitioned into four 1x1 pixels having values 8, 1, 2 and -1. These are coded respectively as binary value 1 with sign bit 0, and binary values 0,0 and 0 without sign bits.

Finally, the bottom right sub-region 716 having values -2, -2, -3, and -3 is coded. A zero (0) is output for each of currentBitNumber = 7, 6, 5, 4 and 3 since the sub-region 716 is insignificant with respect to these bit numbers. No sign bits are output. Thus, the coded representation is as follows:

11110010000010000011000000000111110000001100000000.

The decoder simply mimics the encoding process to reconstruct the region from the coded representation as depicted in Fig. 7C.

The decoding process can be made "smarter" in a number of ways. One such a "smarter" way is depicted in Fig. 7D. In this case, the magnitude of the non-zero coefficients is each increased by half of 2 to the power of minBitNumber. This is depicted in Fig. 7D. In this manner, the "smart" decoding processing generally reduces the mean square error between the decoded and the original coefficients. Still further, the encoder can alternatively perform this (type of) operation, thereby leaving the decoder to use the simplest depicted in Fig. 7C.

1.4 Encoding Process of Second SWEET Image Compression Method

A coding process according to an alternate method is hereinafter described with reference to Figs. 9 to 12.

A discrete wavelet transform of an entire digital image can be performed on a block-by-block basis. The result of the transformation upon each block is a set of coefficients, which are essentially equivalent to a set of spatially corresponding coefficients of a discrete wavelet transform of the entire image. For example, from a predetermined set of coefficients of a DWT for an entire image, a portion or block of the digital image can be reproduced to a specified detail. Selecting the predetermined set of coefficients from the frequency domain amounts substantially to representing the corresponding portion of a digital image (the block) from the spatial domain. A block based DWT of a digital image can be performed by decomposing an image into a plurality of blocks and applying the transform to each block independently, thereby substantially evaluating those DWT coefficients relevant to the current spatial location. The advantage of adopting a block-based transform approach is that a block can be subsequently encoded with minimal interaction (substantially independent) from another block of the image. Block-based techniques are inherently memory localized and therefore are generally efficient when implemented using computer systems.

Fig. 9 is a flow diagram illustrating the block-based encoding process according to the second encoding method. Processing commences at step 902. In step 904, a header is output. This information preferably includes the image height and

width, the block size, the number of levels of the DWT, and two coding parameters maxBitNumber and minBitNumber. Optionally, more or less header information may be used depending upon the application.

The coding parameter maxBitNumber can be selected in a variety of ways. If the block DWT is performed on all image blocks prior to coding of any of them, the maxBitNumber can be chosen to be the MSB number of the largest coefficient across all DWT blocks. For example, if the largest coefficient is 10000001 (decimal value 129), the maxBitNumber is set to 7 since the MSB is bit number 7. Alternatively, a deterministic bound can be used which is determined by the transform and the resolution of the input image. For example, with an 8-bit input image (level shifted to 7-bits plus sign) and the Haar transform, the largest MSB is bounded by J+7 where J is the number of levels of the DWT. If the blocks are small, the selection of this parameter can have a significant effect on compression. In some instances, more sophisticated ways of selecting maxBitNumber may be employed. However, this depends upon the specific application.

The parameter minBitNumber determines the compression ratio versus quality trade off and can be varied. For example, for nearly orthogonal transforms, a value of 3 provides adequate image quality for 8-bit, grey-scale or 24-bit, RGB images.

In step 906, the image is decomposed into blocks (or an image block is formed). The image is decomposed preferably into overlapping blocks. However, non-overlapping blocks may be employed. The block of coefficients can be as large as the whole original image, or as small as a block of 8 x 8 coefficients (for a three-level transform). For low memory applications, a block that is as small as possible may be employed. Generally, a block size of 16 coefficients is sufficient for higher levels of compression with a three or four level DWT. A block size of 8 x 8 coefficients with a three-level DWT can maintain good coding efficiency by employing differential pulse code modulation (DPCM) on the DC coefficient of each block.

In step 908, each block is level shifted and the transform is performed. Preferably, a DWT is employed. The image values are level shifted (for example, by 128 for an 8-bit image) to reduce or eliminate any undue mean bias, and each spatial block of the image is transformed. For a DWT, usually some knowledge of the block surrounding the current block is needed (and similarly for the inverse DWT), although this is not strictly required.

In step 910, the block is coded using the maxBitNumber and minBitNumber parameters. Processing terminates in step 912.

Step 910 for coding a block is illustrated in detail in the flow diagram of Fig. 10. The inputs to the block coding process of Fig. 10 include the currentBitNumber and the minBitNumber parameters. With reference to step 910 of Fig. 9, the

maxBitNumber is input as the currentBitNumber parameter. Processing commences in step 1002. In decision block 1004, a check is made to determine if the currentBitNumber is less than the minBitNumber. If decision block 1004 returns true (yes), processing continues at step 1006. In step 1006, execution returns to the calling process, thereby indicating that every coefficient in the block has an MSB number less than the minBitNumber. Otherwise, if decision block 1004 returns false (no), processing continues at decision block 1008.

In decision block 1008, a check is made to determine if a current block is significant. If decision block 1008 returns false (no), processing continues at step 1010. In step 1010, a zero (0) is output in the coded representation and the currentBitNumber is decremented, that is, the next lower bit plane is selected. Processing then continues at decision block 1004. Otherwise, if decision block 1008 returns true (yes) processing continues at step 1012.

Decision blocks 1004 and 1008 along with step 1010 enable the process to find the MSB number of the largest coefficient in the block. A block is insignificant with respect to the currentBitNumber if the MSB number of every coefficient in the block is less than the currentBitNumber. This is repeated until the bitplane of the block is significant or the currentBitNumber is less than the minBitNumber.

In step 1012, a one (1) is output in the coded representation to indicate the bitplane is significant. In step 1014, the DC subband is coded. In step 1016, the block detail is coded using the parameters J, currentBitNumber and minBitNumber. In step 1018, execution returns to the calling procedure. Thus, given that the block is significant, steps 1012, 1014 and 1016 are carried out to use the (generalized) quadtree segmentation to find all coefficients with an MSB number greater than the minBitNumber. If the block is significant, it is partitioned into two "sub-blocks": the DC subband coefficients and the block consisting of the remaining coefficients, referred to as the "block detail" for level J since it represents the high frequency information about the block of level J at all lower levels.

Step 1014 of Fig. 10 for coding the DC subband is illustrated in detail by the flow diagram of Fig. 12. That is, Fig. 12 shows the process of coding a subband or sub-block using currentBitNumber and minBitNumber parameters. In step 1202, processing commences. In decision block 1204, a check is made to determine if the currentBitNumber is less than the minBitNumber. If decision block 1204 returns true (yes), processing continues at step 1206. In step 1206, execution returns to the calling procedure. Otherwise, if decision block 1204 returns false (no), processing continues at decision block 1208.

In decision block 1208 a check is made to determine if the (subband) block size is 1 x 1 pixels. If decision block 1208 returns true (yes), processing continues at step

1210. In step 1210, the 1 x 1 pixel is coded. This involves outputting the bits between the currentBitNumber and the minBitNumber, inclusive, followed by a sign bit if necessary. Processing then returns to the calling procedure in step 1212. Otherwise, if decision block 1208 returns false (no), processing continues at decision block 1214.

In decision block 1214, a check is made to determine if the (subband) block is significant. If decision block 1214 returns false (no), processing continues at step 1216. In step 1216, a zero (0) is output in the coded representation and the currentBitNumber is decremented. Processing then continues at decision block 1204. Otherwise, if decision block 1214 returns true (yes), processing continues at step 1218.

In step 1218, a one (1) is output in the coded representation to indicate that the (subband) block is significant. In step 1220, the (subband) block is partitioned into four sub-blocks. In step 1222, each sub-block is coded using the parameters currentBitNumber and minBitNumber, by means of a recursive call to the process of Fig. 12. In step 1224, execution returns the calling procedure.

Thus, in the process of Fig. 12, a subband or sub-block thereof is coded. The largest MSB number is isolated as before. If the sub-block consists of only one pixel, it is coded as a single coefficient. Otherwise, the currentBitNumber is decremented and a zero (0) is output in the coded representation until the currentBitNumber is less than the minBitNumber, or the subband (sub-block) is significant. If the subband (sub-block) is significant, it is partitioned into four (as close to equal as possible) sub-block, and these are coded in turn. A single coefficient, for example the DC coefficient, is encoded by outputting the coefficient bits from the currentBitNumber to the minBitNumber. Again, the sign is preferably only output if some of the coefficient bits are non-zero.

Step 1016 of Fig. 10 for coding block detail is illustrated by the flow diagram of Fig. 11. In step 1102, processing commences. In decision block 1104, a check is made to determine if the currentBitNumber is less than the minBitNumber. If decision block 1104 returns true (yes), execution returns to the calling procedure in step 1106. Otherwise, if decision block 1104 returns false (no), processing continues at decision block 1108.

In decision block 1108, a check is made to determine if the block (detail) is significant. If decision block 1108 returns false (no), processing continues at step 1110. In step 1110, a zero (0) is output in the coded representation and the currentBitNumber is decremented. Processing then continues at decision block 1104. Otherwise, if decision block 1108 returns true (yes), processing continues at step 1112.

In step 1112, a one (1) is output in the coded representation to indicate that the block (detail) is significant. In step 1114, each of the high-low (HL), low-high (LH), and high-high (HH) frequency subbands is coded. The HL, LH, and HH frequency subbands of each resolution are commonly referred to as AC subbands. Each of these

subbands is coded in accordance with the process of Fig. 12. In step 1116, the block detail is coded using the parameters J-1, currentBitNumber and minBitNumber (if the block detail exists) by means of a recursive call to the process illustrated in Fig. 11. Execution returns to the calling procedure in step 1118.

Thus, the block detail for level J is processed to first isolate the MSB number of the largest coefficient. This is done by decrementing the currentBitNumber and outputting zeros until the block is significant. The block is then partitioned into the three high frequency subbands at level J and the block detail for level J-1 (if J-1 is greater than 0). This partitioning approach is motivated by the so-called 1/f type spectral models.

The decoding process for the second method can be implemented by mimicking the coding process described with reference to Figs. 9 to 12.

The encoding and decoding methods and apparatuses represent digital image data in an efficient and flexible manner, in which the representation is suitable for storing and/or transmitting images. The encoding techniques can be used generally to represent an array of transform coefficients, and to provide an efficient representation by representing an image in the discrete wavelet transform domain. In particular, the methods and apparatuses represent (or code) leading zeros of blocks of transform coefficients obtained from an input image. The techniques are efficient in terms of offering a good reproduction of the original image for a given size code and offering fast decoding. Further, the techniques are flexible in that coefficients obtained from a linear transformation are encoded independently without the use of entropy coding. The advantageous aspects of the methods include the depth first nature of the coding. Further, in the case of coding subbands, the advantageous aspects of the method include hierarchical coding of each subband separately.

2. Preferred Embodiment(s) of Method

The preferred embodiment utilises the wavelet transform process and proceeds initially by means of a wavelet transform of image data. A description of the wavelet transform process is given in many standard texts and in particular the aforementioned book by Stollnitz et. al. An overview of the wavelet process will now be described with reference to the accompanying drawings.

Referring initially to Fig. 13, an original image 1 is transformed utilising a Discrete Wavelet Transform (DWT) into four subimages 3-6. The subimages or subbands are normally denoted LL1, HL1, LH1 and HH1. The one suffix on the subband names indicates level 1. The LL1 subband is a low pass decimated version of the original image.

The wavelet transform utilised can vary and can include, for example, Haar basis functions, Daubechies basis functions etc. The LL1 subband is then in turn utilised and a second Discrete Wavelet Transform is applied as shown in Fig. 14 giving subbands LL2 (8), HL2 (9), LH2 (10), HH2 (11). This process is continued for example as illustrated in Fig. 15 wherein the LL4 subband is illustrated, the LL4 band decomposition process being referred to as an octave band filter bank with the LL4 subband being referred to as the DC subband. Obviously, further levels of decomposition can be provided depending on the size of the input image.

Each single level DWT can in turn be inverted to obtain the original image. Thus a J-level DWT can be inverted as a series of J-single level inverse DWT's.

An image coding hierarchically can proceed by coding the DC subband. Then, the remaining subbands are coded in order of decreasing level. That is for a 4 level DWT, the subbands at level 4 are coded after the DC subband (LL4). That is the HL4, LH4 and HH4 subbands. The subbands at level 3 (HL3, LH3, and HH3) are then coded, followed by those at level 2 (HL2, LH2 and HH2) and then level 1 (HL1, LH1 and HH1).

With standard images, the encoded subbands normally contain the "detail" information in an image. Hence, they often consist of a sparse array of values and substantial compression can be achieved by quantisation of the subbands and efficient encoding of their sparse matrix form.

An overview of the encoding process is as illustrated by the flow chart 20 of Fig. 16. The encoded process begins by applying a discrete wavelet transform 22 to the image data to produce the usual subbands. Next, the subbands are looped through in hierarchical order 23 from the lowest frequency subband to the highest. For each subband three steps 24-26 are performed wherein the step 24 includes determining the best quadtree and associated quantisation factors for the subband. Next, each region is quantised as specified by the quadtree with the appropriate quantiser 25. Next, the quantised subband values and the appropriate quadtree information is encoded 26.

Turning now to Fig. 17, there is illustrated the decoding process. Each of the subbands is treated in order with the subbands being looped through 31 and the inverse processes 32-34 are applied to the subbands. The first step 32 includes decoding the quantisation and quadtree information. Next, the quantised coefficients are decoded 32 before an inverse quantisation process is applied 34. The resultant data is then inverse discrete wavelet transformed 35 so as to produce the original data.

The core portion of the preferred embodiment is the process of determining the best quadtree and associated quantisation factors for the subband 24. Each subband is partitioned into variable sized regions using a quadtree structure and associated with each region, or leaf within the quadtree, is a quantisation factor. The utilisation of

quadtrees can proceed along standard lines. For a full description of quadtrees and alternative data structures that may be suitable, reference is made to the standard survey article entitled "The Quadtree and Related Hierarchical Data Structures" by Hanan Samet, published in Computer Surveys, vol. 16, no. 2, June 1984 at pages 187-260. In the preferred embodiment, each region is uniformly quantised with a quantisation parameter represented as an index value. The index values for each subband are then encoded along with the quadtree structure and the quantisation data associated with each leaf of the quadtree.

Although many different encoding processes can be utilised, in the preferred embodiment, the coding of the quantisation data can be achieved utilising the SWEET methodology as disclosed in Australian provisional patent specification No. PO4728 and described herein and hereinafter called SWEET coding. As will become evident, the present invention differs from previous SWEET coding in that it utilizes dynamic quantisation.

In order to be able to vary the quantisation spatially within a subband, a quadtree structure is utilised to divide up the image. The quadtree can be used to represent a certain partition of a region into several variable sized subregions. For example, given a rectangular region or array of coefficients, it is possible to partition the region into four subregions, in accordance with standard quadtree techniques. Within each region, it is possible to recursively partition or leave the region "as is" depending on a predetermined criterion. For example, Fig. 18 illustrates an example quadtree structure, the structure of which will be readily familiar to those familiar with quadtrees. In this respect, the region 40 has been partitioned into four subregions with only the regions 41 and 42 being "sub-partitioned". With the bottom right region 42 being further sub-partitioned 43.

With each leaf node of the quadtree, a quantisation parameter is determined. A different quantisation factor can be provided for each subregion. The structure of the quadtree 40 and associated quantisation parameters are coded into the compressed image bit stream. The decoding process utilises this information to inverse quantise each subregion with an appropriate quantisation factor.

A quadtree can be represented as a sequence of binary partitioning decisions. A "1" can be used to represent a partition of a region, while a "0" can be used to represent a leaf node, or the fact that a region is not partitioned. Using a depth first approach to representation, immediately following a 1 bit describing a partition of a region are the binary partitioning decisions of the corresponding subregions. The subregion partitioning decisions can be encoded in the order of top left, top right, bottom left and bottom right. Thus the quadtree 40 of Fig. 18 can be represented by the binary sequence 11000000101000000. The first "1" represents a partition of the

whole (original) region 40 into 4 subregions. The next "1" represents the partition of the top-left quadrant 41 of the original region into 4 subregions. The next four "0"'s represent the fact that these latter subregions of region 41 are leaf nodes, and not partitioned. The following two 0's represent that the top right and bottom left quadrants of the original region are leaf nodes. Finally the remaining bits 101000000 represent the partitioning structure of the bottom right quadrant 42 of the original region.

With each leaf node of the quadtree is an associated quantiser. It is possible to simply code a quantisation parameter for each leaf node in the order in which the leaf node occurs in the quadtree. Having decoded the quadtree representation the decoder can then calculate the ordering and decode each quantisation parameter in sequence.

The preferred process of quantisation includes using uniform quantisation with a dead-zone. The discrete wavelet transform coefficients are quantised to integer values. Let c represent a coefficient value, d its quantised value and let q be the quantisation parameter for the region in which the coefficient lies. Then the quantisation can be defined as,

$$d = fix\left(\frac{c}{q}\right)$$

where fix is defined by,

$$fix(x) = \begin{cases} \lfloor x \rfloor & x \geq 0 \\ \lceil x \rceil & x \leq 0 \end{cases}$$

and $\lfloor\ \rfloor$ is the usual "floor" round down to nearest integer operator and $\lceil\ \rceil$ is the usual "roof" round up to nearest integer operator. The quantisation parameter q is the quantisation factor. At the encoder each coefficient in a subband is quantised to an integer value using this equation.

The inverse quantisation is given by,

$$\hat{c} = q \times d + sign(d) \times \frac{q}{2}$$

where,

$$sign(d) = \begin{cases} -1 & d < 0 \\ 0 & d = 0 \\ 1 & d > 0 \end{cases}$$

At the decoder each coefficient is inverse quantised using this inverse quantisation equation.

For a given rate, of say R bits per pixel (bpp), the best quantisation quadtree and associated quantisation factors can be defined as the quadtree that results in a compressed image with minimum distortion. That is compressing the image to R bpp, or to a lower rate, results in a higher distortion when using any other quadtree. The set

of possible quantisation factors is presumed to be a fixed finite set. In the present case a set of 16 possible quantisation scale factors ranging from 1 to 16 have been used. With 16 different possible quantisation scale factors, a simple code will require 4 bits to determine a given scale factor.

The method of finding the best such quadtree is naturally found using the Lagrange multiplier approach for constrained optimisation. That is for a given $\lambda > 0$, find

$$\min [\text{cost} = d(Q)+\lambda b(Q)] \quad \text{(Eqn1)}$$

where the minimisation is over all quadtrees and associated quantisation factors. The quadtree and associated quantisation factors is represented in the equation as Q. Thus this minimisation is over all possible Q. If the optimum solution to the unconstrained Lagrangian problem is given by Q* then Q* is the optimum solution to the constrained solution in the case where the given rate is b(Q*). To find the optimum solution for a given rate it is possible to simply vary λ using a bisection method until b(Q*) is sufficiently close to the desired rate R.

The method used to perform the minimisation in Eqn1 for a given image region is illustrated in the following pseudo code:

```
[bits, distortion, nmax] = findQuantisationQuadtree (region, n, λ)
{
    */
    **Find the optimum fixed quantisation factor for the region
    */
    [bits, distortion, q] = findOptimumFixedQuantisation (region, n, λ);
    set fixedQuantisationCost = distortion + λ bits;
    /*
    **Find the cost of using quadtree variable quantisation on the region
    **qmin is the smallest quantisation factor
    */
    set nmax = largest bit number in region quantised with qmin;
    partition region into 4 subregions;
    [bits1, distortion], nmax1] = findQuantisationQuadtree region1, nmax, λ);
    [bits2, distortion2, nmax2] = findQuantisationQuadtree region2, nmax, λ);
    [bits3, distortion3, nmax3] = findQuantisationQuadtree region3, nmax, λ);
    (bits4, distortion4, nmax4] = findQuantisationQuadtree region4, nmax, λ);
    set bitSavings = 3 x (nmax - max(nmax1, nmax2, nmax3, nmax4);
    set variableQuantisationCost = distortion1 + distortion2 +
```

```
        distortion3 + distortion4 + λ (bits1 + bits2 + bits3 + bits4 +
        (n-nmax+1) - bitSavings + 1);
        /*
        **Find the better approach - variable or fixed quantisation.
        */
        if variableQuantisationCost < fixedQuantisationCost
        {
            distortion = distortion1 + distortion2 + distortion3 + distortion4;
            bits = bits1 + bits2 + bits3 + bits4 + (n-nmax+1) - bitSavings + 1;
            nmax = max(nmax1, nmax2, nmax3, nmax4);
        }
        else // Fixed quantisation is better, (return parameters are already set correctly)
        {
            output region and quantisation factor q
        }
    }

    [bits, distortion, q] = findOptimumFixedQuantisation (region, n, λ)
    {
        /*
        **Find the quantisation factor with the least Lagrangian cost
        */
        set cost = infinity;
        loop through the quantisation factors qi
        {
            quantise region with quantisation factor qi;
            bitsQ = SWEET code (region, n);
            set distortionQ to the quantisation distortion;
            if distortionQ + λbitsQ < cost
            {
                cost = distortionQ + λbitsQ
                bits = bitsQ;
                distortion = distortionQ;
                q = qi;
            }
        }
        /*
        **Add in the number of bits used to code the fixed quantisation factor and the
```

```
        **termination bit for the quantisation quadtree
        */
        set bits = bits + numQuantisationFactorBits + 1;
    }
```

In this code the set of (16) possible quantisation parameters is implicit.

In the findOptimumQuadtree function above, the optimum fixed quantisation factor to use for the whole region and its associated cost is calculated by calling the findOptimumFixedQuantisation function. Then the cost associated with using a quadtree variable quantisation on the region is calculated. This is performed by partitioning the region into four sub regions and recursively calling the findOptimumQuadtree function on each subregion. The fixed and variable quantisation costs are then compared and the solution with the minimum cost is selected. If fixed quantisation is selected then the region and associated quantisation factor are output. This output can then be used later to code the quadtree and associated quantisation factors, and to quantise the image regions prior to coding.

The $(n- n_{max}+1)$ - bitSavings number of bits factor in the calculation of variableQuantisationCost in the function findOptimumQuadtree, is the number of extra bits that the SWEET coding method requires to code the four subregions given the bit number parameter n. The final extra bit added to the number of bits factor in the calculation of variableQuantisationCost is a 1 bit used to indicate the partition of the region in the quantisation quadtree.

In the findOptimumFixedQuantisation, the SWEET coding method is used to calculate the number of bits used to code a quantised region. The best quantisation factor is then found as the one that minimises the Lagrangian cost, distortion + λbits. Finally to the total number of bits used to code the region, with the best quantisation factor, is added numQuantisationFactorBits, which is simply the number of bits required to code the quantisation factor (in our case this is 4 bits), and a single bit representing the 0 terminating bit in the quadtree indicating that the region is a leaf node. 0

2.1 Preferred Embodiment of Apparatus(s)

The encoding and/or decoding processes are preferably practiced using a conventional general-purpose computer, such as the one shown in Fig. 8, wherein the processes of Fig. 16 or 17 may be implemented as software executing on the computer. In particular, the steps of the encoding and/or decoding methods are effected by instructions in the software that are carried out by the computer. The software may be divided into two separate parts; one part for carrying out the encoding and/or decoding methods; and another part to manage the user interface between the latter and the user.

The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for encoding and/or decoding coded representations of digital images in accordance with the embodiments of the invention.

The computer system 800 consists of the computer 802, a video display 816, and input devices 818, 820. In addition, the computer system 800 can have any of a number of other output devices including line printers, laser printers, plotters, and other reproduction devices connected to the computer 802. The computer system 800 can be connected to one or more other computers via a communication interface 808c using an appropriate communication channel 830 such as a modem communications path, a computer network, or the like. The computer network may include a local area network (LAN), a wide area network (WAN), an Intranet, and/or the Internet

The computer 802 itself consists of a central processing unit(s) (simply referred to as a processor hereinafter) 804, a memory 806 which may include random access memory (RAM) and read-only memory (ROM), input/output (IO) interfaces 808a, 808b & 808c, a video interface 810, and one or more storage devices generally represented by a block 812 in Fig. 8. The storage device(s) 812 can consist of one or more of the following: a floppy disc, a hard disc drive, a magneto-optical disc drive, CD-ROM, magnetic tape or any other of a number of non-volatile storage devices well known to those skilled in the art. Each of the components 804 to 812 is typically connected to one or more of the other devices via a bus 814 that in turn can consist of data, address, and control buses.

The video interface 810 is connected to the video display 816 and provides video signals from the computer 802 for display on the video display 816. User input to operate the computer 802 can be provided by one or more input devices 808b. For example, an operator can use the keyboard 818 and/or a pointing device such as the mouse 820 to provide input to the computer 802.

The system 800 is simply provided for illustrative purposes and other configurations can be employed without departing from the scope and spirit of the invention. Exemplary computers on which the embodiment can be practiced include IBM-PC/ATs or compatibles, one of the Macintosh (TM) family of PCs, Sun Sparcstation (TM), or the like. The foregoing is merely exemplary of the types of computers with which the embodiments of the invention may be practiced. Typically, the processes of the embodiments, described hereinafter, are resident as software or a program recorded on a hard disk drive (generally depicted as block 812 in Fig. 8) as

the computer readable medium, and read and controlled using the processor 804. Intermediate storage of the program and pixel data and any data fetched from the network may be accomplished using the semiconductor memory 806, possibly in concert with the hard disk drive 812.

In some instances, the program may be supplied to the user encoded on a CD-ROM or a floppy disk (both generally depicted by block 812), or alternatively could be read by the user from the network via a modem device connected to the computer, for example. Still further, the software can also be loaded into the computer system 800 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

The method of coding and/or decoding may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of the encoding and decoding. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

The foregoing only describes a small number of embodiments of the present invention, however, modifications and/or changes can be made thereto by a person skilled in the art without departing from the scope and spirit of the invention. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

4. Brief Explanation of the Drawings

Fig. 1 is a high-level block diagram illustrating the image representation technique described in the herein-mentioned patent application;

Fig. 2 is a diagram illustrating partitioning described in the herein-mentioned patent application;

Fig. 3 is a flow diagram illustrating the method of representing, or encoding, an image described in the herein-mentioned patent application;

Fig. 4 is a detailed flow diagram illustrating the step of coding a region in Fig. 3;

Fig. 5 is a flow diagram illustrating the method of decoding a coded representation of an image produced in accordance with the method Fig. 3;

Fig. 6 is a detailed flow diagram illustrating the step of decoding a region in Fig. 5;

Figs 7A to 7D are diagrams illustrating the processing of a two-dimensional, eight-coefficient region in accordance with the encoding and decoding method of Figs. 3 to 6;

Fig. 8 is a block diagram of a general purpose computer;

Figs. 9 to 12 are flow diagrams illustrating an alternate method representing, or encoding, an image described in the herein-mentioned patent application;

Figs. 13-15 illustrate the process of wavelet transforming image data;

Fig. 16 illustrates a flow chart of the encoder operation;

Fig. 17 illustrates the operation of the decoder; and

Fig. 18 illustrates one particular form of quadtree partition.

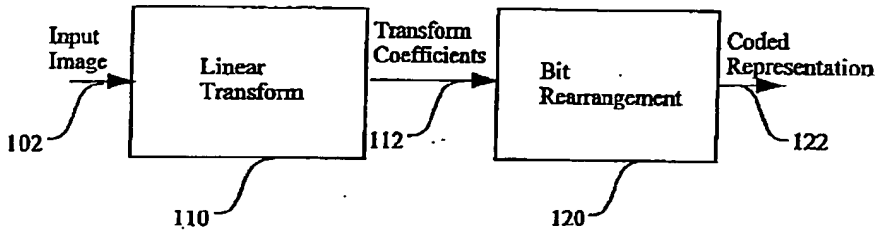

Input
Image
102
Linear
Transform
110
Transform
Coefficients
112
Bit
Rearrangement
120
Coded
Representation
122

Fig. 1

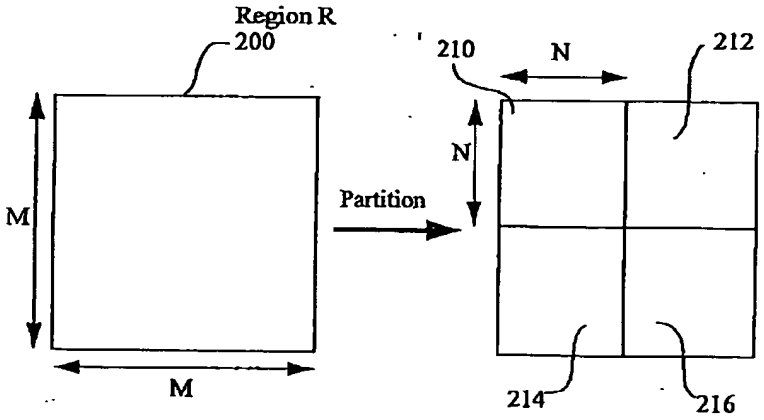


Fig. 2

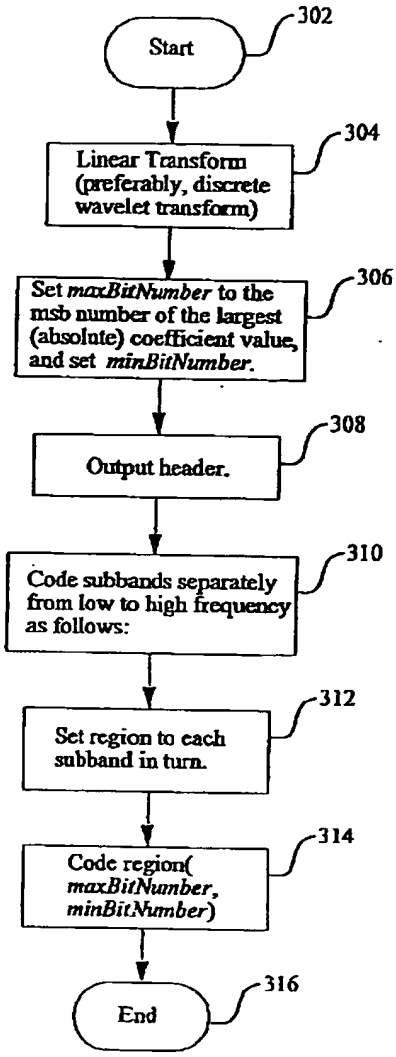

Start
302
Linear Transform (preferably, discrete wavelet transform)
304
Set *maxBitNumber* to the msb number of the largest (absolute) coefficient value, and set *minBitNumber*.
306
Output header.
308
Code subbands separately from low to high frequency as follows:
310
Set region to each subband in turn.
312
Code region(*maxBitNumber*, *minBitNumber*)
314
End
316

Fig. 3

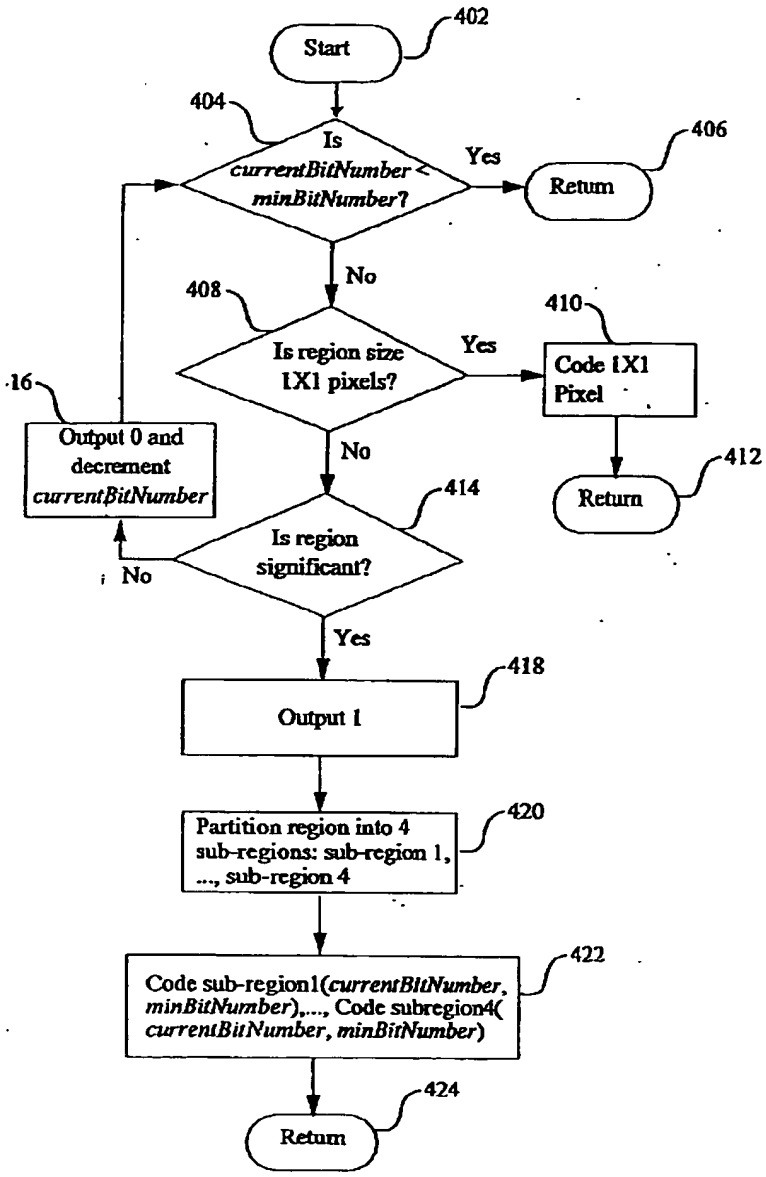

Start
402
404
Is currentBitNumber < minBitNumber?
Yes
406
Return
No
408
Is region size 1X1 pixels?
Yes
410
Code 1X1 Pixel
412
Return
No
414
Is region significant?
No
16
Output 0 and decrement currentBitNumber
Yes
418
Output 1
420
Partition region into 4 sub-regions: sub-region 1, ..., sub-region 4
422
Code sub-region1(currentBitNumber, minBitNumber),..., Code subregion4(currentBitNumber, minBitNumber)
424
Return

Fig. 4

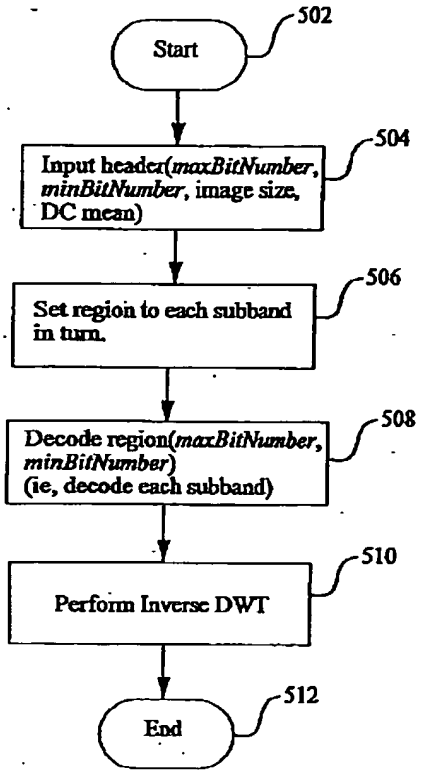

502
Start
504
Input header(maxBitNumber, minBitNumber, image size, DC mean)
506
Set region to each subband in turn.
508
Decode region(maxBitNumber, minBitNumber) (ie, decode each subband)
510
Perform Inverse DWT
512
End

Fig. 5

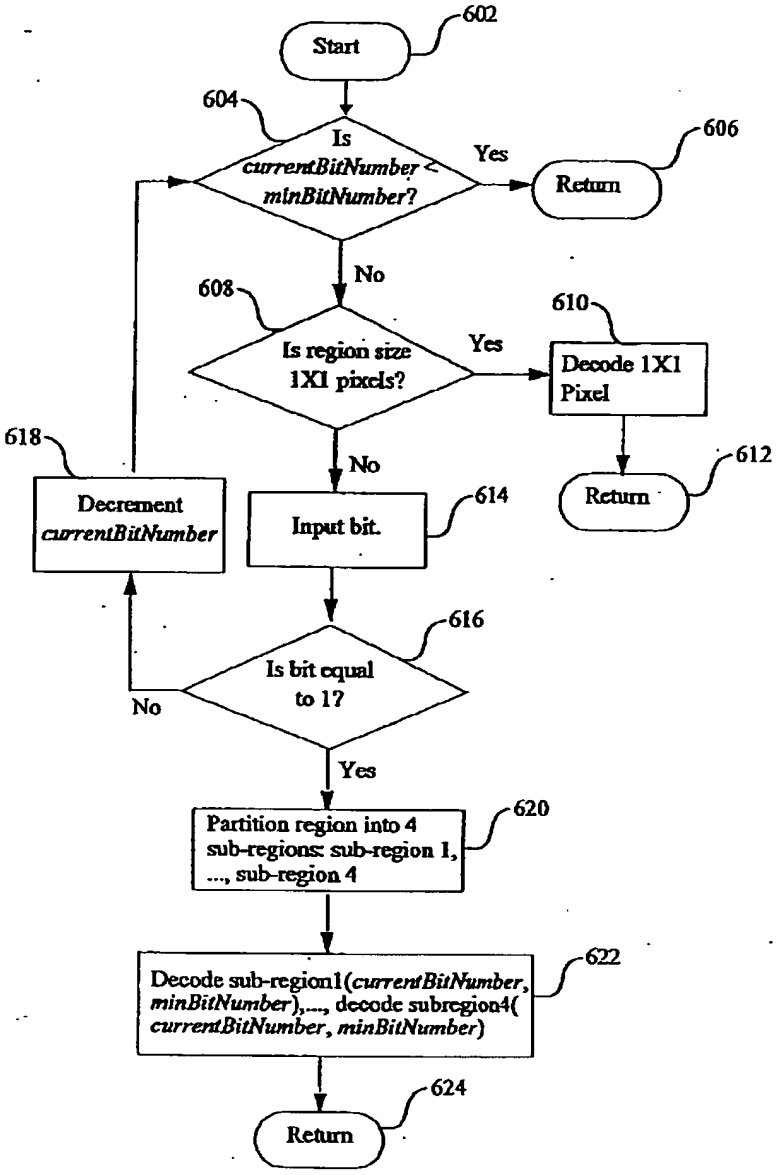

Start
602
604
Is currentBitNumber < minBitNumber?
Yes
606
Return
No
608
Is region size 1X1 pixels?
Yes
610
Decode 1X1 Pixel
612
Return
No
614
Input bit.
618
Decrement currentBitNumber
616
Is bit equal to 1?
No
Yes
620
Partition region into 4 sub-regions: sub-region 1, ..., sub-region 4
622
Decode sub-region1(currentBitNumber, minBitNumber),..., decode subregion4(currentBitNumber, minBitNumber)
624
Return

Fig. 6

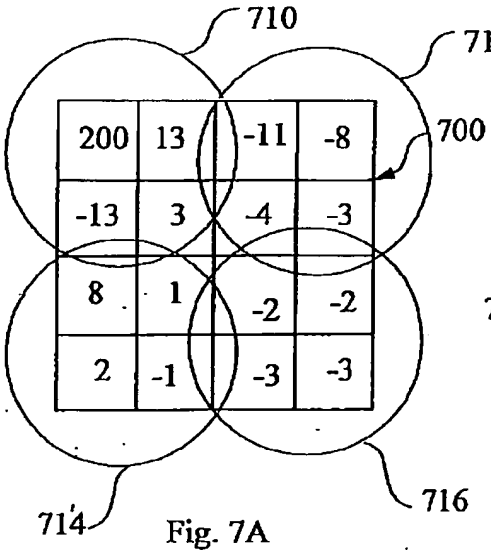


Fig. 7A

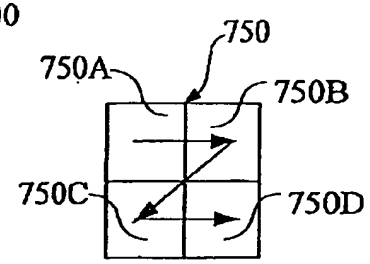


Fig. 7B

200	8	-8	-8
-13	0	0	0
8	0	0	0
0	0	0	0

Fig. 7C

204	12	-12	-12
-12	0	0	0
12	0	0	0
0	0	0	0

Fig. 7D

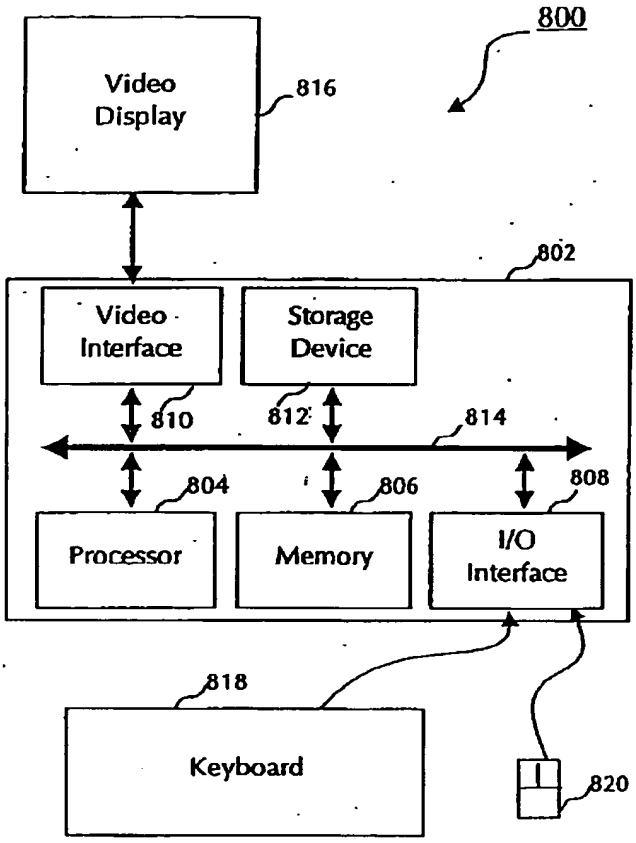

800
Video Display
816
802
Video Interface
Storage Device
810
812
814
804
806
808
Processor
Memory
I/O Interface
818
Keyboard
820

Fig. 8

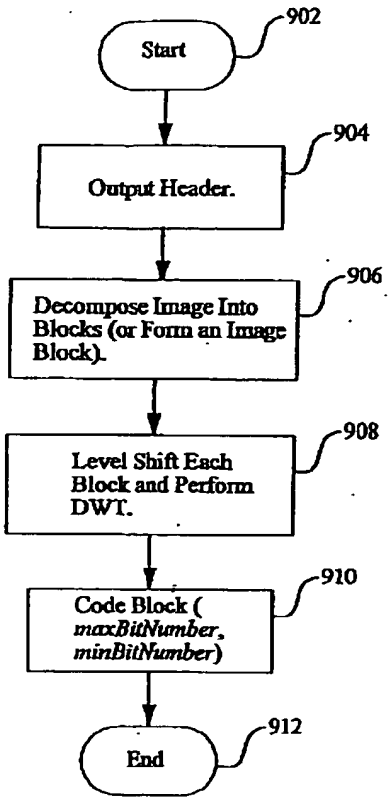

902
Start
904
Output Header.
906
Decompose Image Into Blocks (or Form an Image Block).
908
Level Shift Each Block and Perform DWT.
910
Code Block (maxBitNumber, minBitNumber)
912
End

Fig. 9

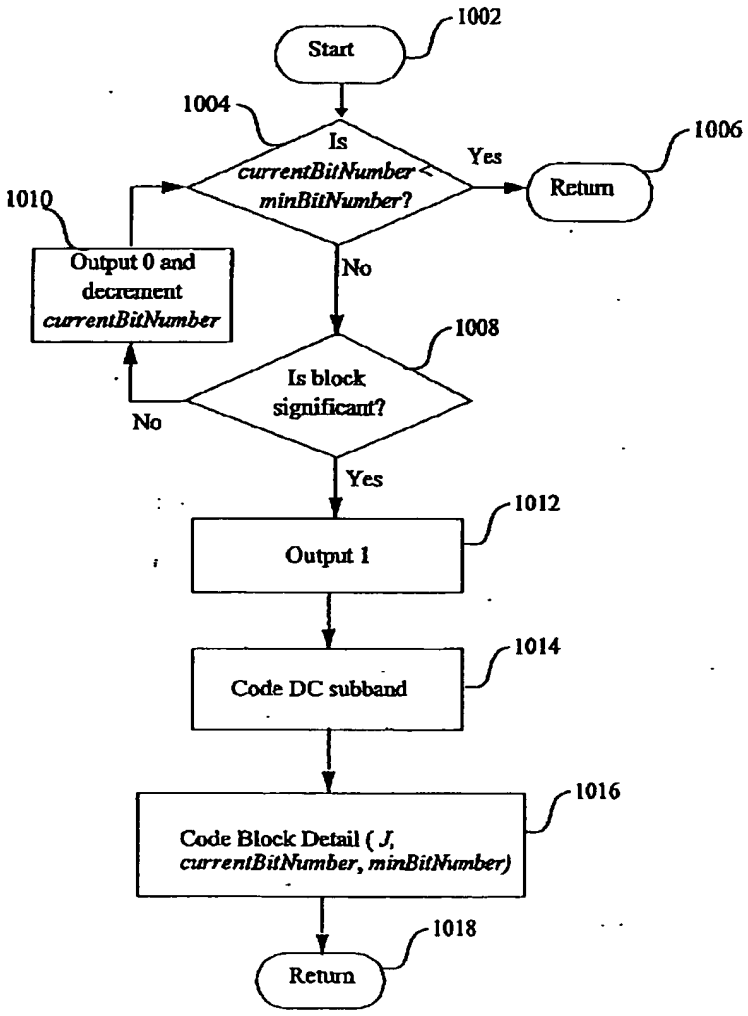

Start
1002
1004
Is
currentBitNumber <
minBitNumber?
Yes
1006
Return
No
1010
Output 0 and
decrement
currentBitNumber
1008
Is block
significant?
No
Yes
1012
Output 1
1014
Code DC subband
1016
Code Block Detail (J,
currentBitNumber, minBitNumber)
1018
Return

Fig. 10

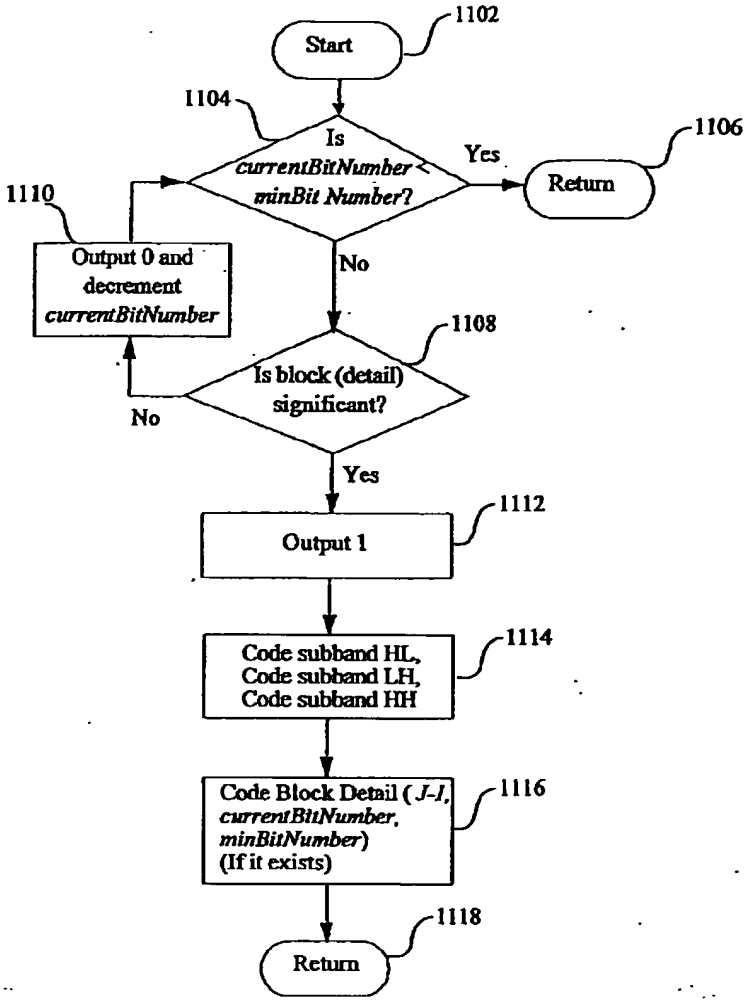

Start
1102
1104
Is
currentBitNumber <
minBit Number?
Yes
1106
Return
1110
Output 0 and
decrement
currentBitNumber
No
1108
Is block (detail)
significant?
No
Yes
1112
Output 1
Code subband HL,
Code subband LH,
Code subband HH
1114
Code Block Detail (J-I,
currentBitNumber,
minBitNumber)
(If it exists)
1116
Return
1118

Fig. 11

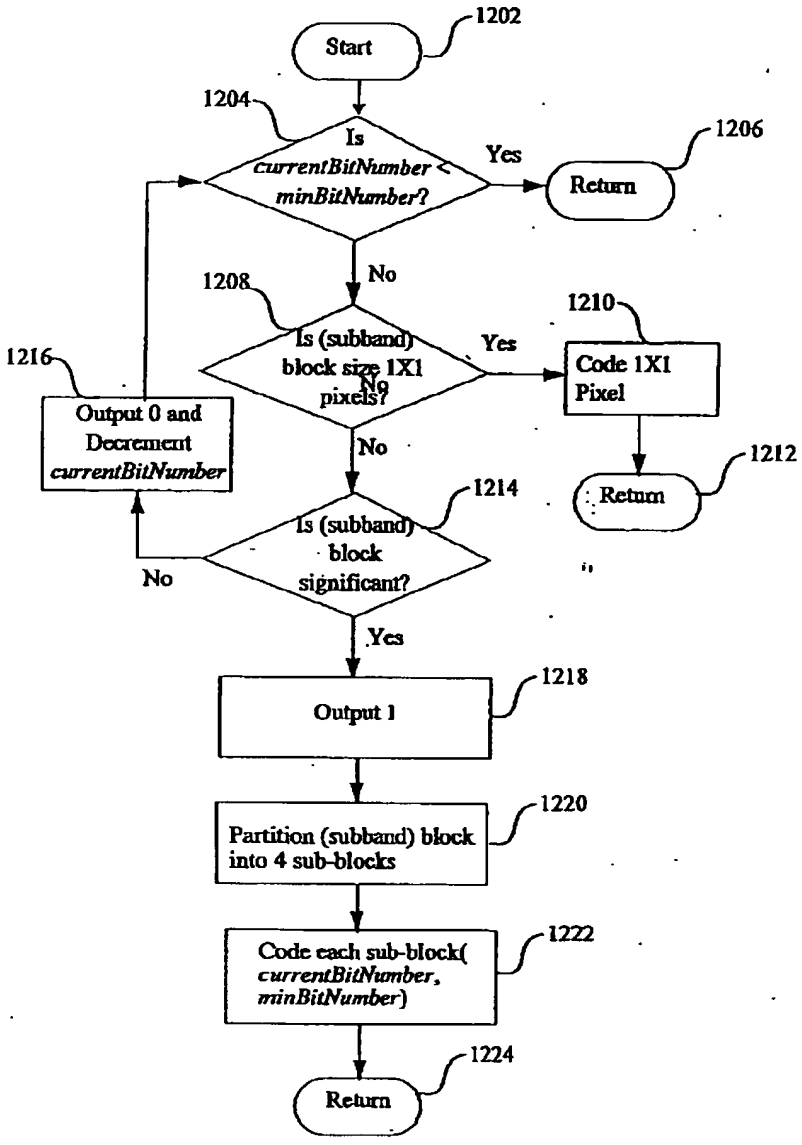

Start
1202
1204
Is currentBitNumber < minBitNumber?
Yes
1206
Return
No
1208
Is (subband) block size 1X1 pixels?
Yes
1210
Code 1X1 Pixel
1212
Return
No
1216
Output 0 and Decrement currentBitNumber
1214
Is (subband) block significant?
No
Yes
1218
Output 1
1220
Partition (subband) block into 4 sub-blocks
1222
Code each sub-block(currentBitNumber, minBitNumber)
1224
Return

Fig. 12

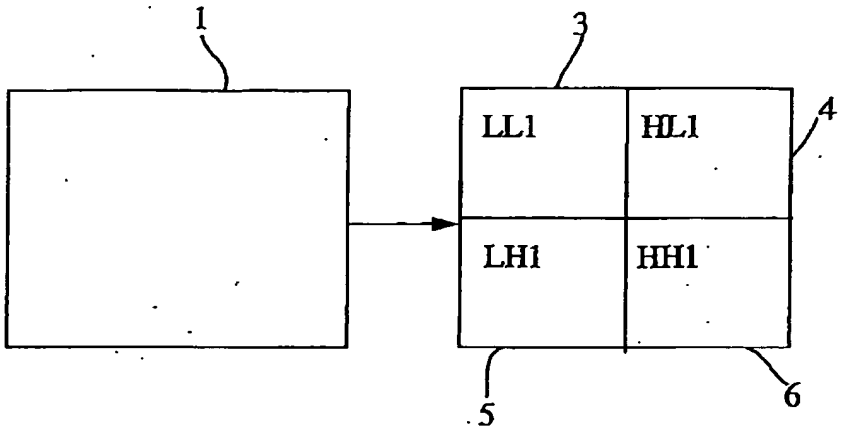


Fig.13

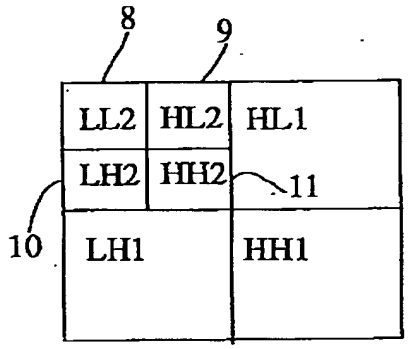


Fig 14

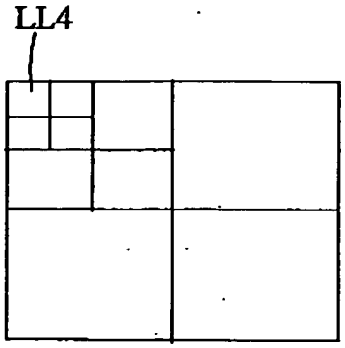


Fig 15

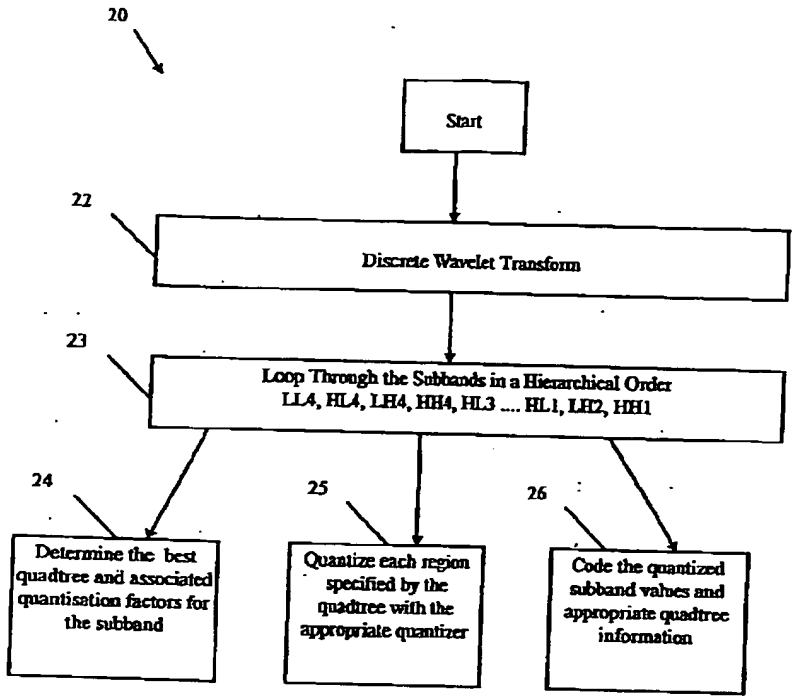

20
Start
22
Discrete Wavelet Transform
23
Loop Through the Subbands in a Hierarchical Order
LL4, HL4, LH4, HH4, HL3 .... HL1, LH2, HH1
24
Determine the best quadtree and associated quantisation factors for the subband
25
Quantize each region specified by the quadtree with the appropriate quantizer
26
Code the quantized subband values and appropriate quadtree information

Fig. 16

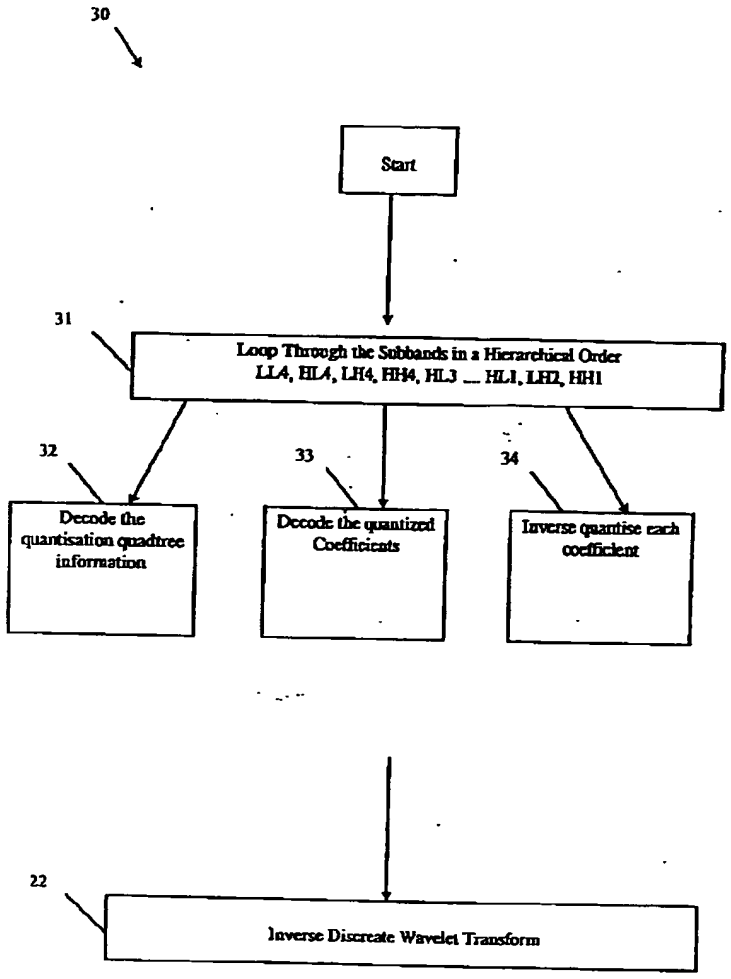

30
Start
31
Loop Through the Subbands in a Hierarchical Order
LLA, HLA, LH4, HH4, HL3 ... HL1, LH2, HH1
32
Decode the quantisation quadtree information
33
Decode the quantized Coefficients
34
Inverse quantise each coefficient
22
Inverse Discreate Wavelet Transform

Fig. 17

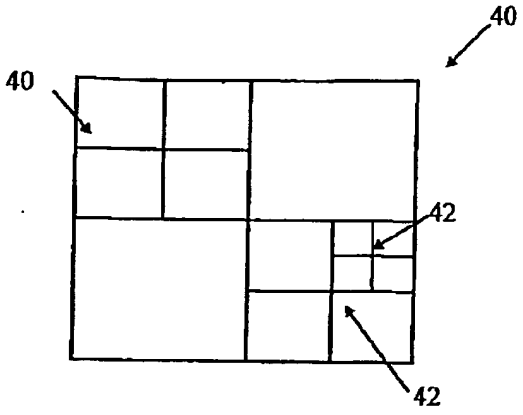


Fig. 18

1. Abstract

A method of compressing digital data is disclosed including the steps of transforming the data utilising a discrete wavelet transform to produce corresponding transformed data; quantising the transformed data utilising a variable quantisation determined by a corresponding quadtree structure wherein each of the quadtree leaf nodes has an associated quantisation factor utilised in the quantising of the transformed data. Preferably, the quadtree is determined to be an optimum in a rate distortion sense and encoded utilising a binary prefix notation followed by a list of quantisation factors. The method of Lagrange multipliers can be utilised to determine the optimum to a predetermined number of bits per data item. The present invention has particular application to image data or to video data and in particular frame difference data.

2. Representative Drawing

Fig. 1